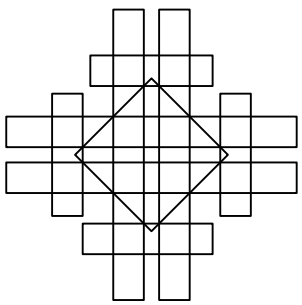
42nd International Symposium on Computational Geometry

SoCG 2026, June 2–5, 2026, New Brunswick, NJ, USA

Edited by
Hee-Kap Ahn
Michael Hoffmann
Amir Nayyeri



LIPICS

Editors

Hee-Kap Ahn
Pohang University of Science and Technology (POSTECH), Korea
heekap@postech.ac.kr

Michael Hoffmann
ETH Zürich, Switzerland
hoffmann@inf.ethz.ch

Amir Nayyeri
Oregon State University, Corvallis, OR, USA
amir.nayyeri@oregonstate.edu

ACM Classification 2012
Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms; Mathematics of computing → Graph algorithms; Mathematics of computing → Combinatorics

ISBN 978-3-95977-418-5

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-418-5.

Publication date
June, 2026

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists all publications of this volume in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

License
This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0): https://creativecommons.org/licenses/by/4.0/legalcode.
In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:
- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.SoCG.2026.0

ISBN 978-3-95977-418-5 **ISSN 1868-8969** **https://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Editorial Board

- Christel Baier (TU Dresden, DE)
- Roberto Di Cosmo (Inria and Université Paris Cité, FR)
- Faith Ellen (University of Toronto, CA)
- Javier Esparza (TU München, DE)
- Holger Hermanns (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)
- Daniel Král' (Leipzig University, DE and Max Planck Institute for Mathematics in the Sciences, Leipzig, DE – Chair)
- Sławomir Lasota (University of Warsaw, PL)
- Meena Mahajan (Institute of Mathematical Sciences, Chennai, IN)
- Chih-Hao Luke Ong (Nanyang Technological University, SG)
- Eva Rotenberg (IT University of Copenhagen, DK)
- Pierre Senellart (ENS, Université PSL, Paris, France)
- Alexandra Silva (Cornell University, Ithaca, US)

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Preface
Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri 0:xiii

Conference Organization
..... 0:xv

Additional Reviewers
..... 0:xix

Regular Papers

Covering and Partitioning Complex Objects with Small Pieces
Anders Aamand, Mikkel Abrahamsen, Reilly Browne, Mayank Goswami, Prahlad Narasimhan Kasthurirangan, Linda Kleist, Joseph S. B. Mitchell, Valentin Polishchuk, and Jack Stade 1:1–1:16

On the Maximum Number of Tangencies Among 1-Intersecting Curves
Eyal Ackerman and Balázs Keszegh 2:1–2:15

Lower Bounding the Gromov–Hausdorff Distance in Metric Graphs
Henry Adams, Sushovan Majhi, Fedor Manin, Žiga Virk, and Nicolò Zava 3:1–3:16

Dynamic Nearest-Neighbor Searching Under General Metrics in $\mathbb{R}^3$ and Its Applications
Pankaj K. Agarwal, Matthew J. Katz, and Micha Sharir 4:1–4:15

Bifunction and Interlevel Delaunay Trifiltrations
Ángel Javier Alonso, Michael Kerber, Tung Lam, Michael Lesnick, and Abhishek Rathod 5:1–5:20

Estimating the Persistent Homology of $\mathbb{R}^n$-Valued Functions Using Function-Geometric Multifiltrations
Ethan André, Jingyi Li, David Loiseaux, and Steve Oudot 6:1–6:18

Computing L_∞ Hausdorff Distances Under Translations: The Interplay of Dimensionality, Symmetry and Discreteness
Sebastian Angrick, Kevin Buchin, Geri Gokaj, and Marvin Künnemann 7:1–7:15

Cauchy's Surface Area Formula in the Funk Geometry
Sunil Arya and David M. Mount 8:1–8:18

Intersection Patterns of Set Systems on Manifolds with Slowly Growing Homological Shatter Functions
Sergey Avvakumov, Marguerite Bin, and Xavier Goaoc 9:1–9:16

Fast Free Resolutions of Bifiltered Chain Complexes
Ulrich Bauer, Tamal K. Dey, Michael Kerber, Florian Russold, and Matthias Söls 10:1–10:21

Locally Correct Interleavings Between Merge Trees
Thijs Beurskens, Tim Ophelders, Bettina Speckmann, and Kevin Verbeek 11:1–11:15

42nd International Symposium on Computational Geometry (SoCG 2026).
Editors: Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Dynamic and Streaming Algorithms for Union Volume Estimation
Sujoy Bhore, Karl Bringmann, Timothy M. Chan, and Yanheng Wang 12:1–12:15

Dynamic Light Spanners in Doubling Metrics
Sujoy Bhore, Jonathan Conroy, and Arnold Filtser 13:1–13:16

Improved Online Hitting Set Algorithms for Structured and Geometric Set Systems
Sujoy Bhore, Anupam Gupta, and Amit Kumar 14:1–14:14

Euclidean Noncrossing Steiner Spanners of Nearly Optimal Sparsity
Sujoy Bhore, Sándor Kisfaludi-Bak, Lazar Milenković, Csaba D. Tóth, Karol Węgrzycki, and Sampson Wong 15:1–15:18

Flip Distance of Non-Crossing Spanning Trees: NP-Hardness and Improved Bounds
Håvard Bakke Bjerkevik, Joseph Dorfer, Linda Kleist, Torsten Ueckerdt, and Birgit Vogtenhuber 16:1–16:18

Fréchet Distance in the Imbalanced Case
Lotte Blank 17:1–17:17

Product Structure and Treewidth of Hyperbolic Uniform Disk Graphs
Thomas Bläsius, Emil Dohse, Deborah Haun, and Laura Merker 18:1–18:17

Splitting Sandwiches Unevenly via Unique Sink Orientations and Rainbow Arrangements
Michaela Borzechowski, Sebastian Haslebacher, Hung P. Hoang, Patrick Schnider, and Simon Weber 19:1–19:16

The Spanning Ratio of the Directed Θ_6-Graph Is 5
Prosenjit Bose, Jean-Lou De Carufel, John Stuart, and Darryl Hill 20:1–20:18

On Minimum Venn Diagrams
Sofia Brenner, Petr Gregor, Torsten Mütze, and Francesco Verciani 21:1–21:18

Disproving Two Conjectures on the Hamiltonicity of Venn Diagrams
Sofia Brenner, Linda Kleist, Torsten Mütze, Christian Rieck, and Francesco Verciani 22:1–22:16

Shortest Paths in Geodesic Unit-Disk Graphs
Bruce W. Brewer and Haitao Wang 23:1–23:15

Single-Criteria Metric $\boldsymbol{r}$-Dominating Set Problem via Minor-Preserving Support
Reilly Browne and Hsien-Chih Chang 24:1–24:17

The Typical Algebraic Shifting of Graphs and Surfaces
Denys Bulavka, Eran Nevo, and Yuval Peled 25:1–25:15

Finding a Fair Scoring Function for Top-k Selection: From Hardness to Practice
Guangya Cai 26:1–26:17

The Squishy Grid Problem
Zixi Cai, Kuowen Chen, Shengquan Du, Arnold Filtser, Seth Pettie, and Daniel Skora 27:1–27:16

Triangulating a Polygon with Holes in Optimal (Deterministic) Time
Timothy M. Chan 28:1–28:13

Charting the Diameter Computation Landscape of Intersection Graphs in 3D and Above
Timothy M. Chan, Hsien-Chih Chang, Jie Gao, Sándor Kisfaludi-Bak, Hung Le, and Da Wei Zheng 29:1–29:15

Computing the Girth of a Segment Intersection Graph
Timothy M. Chan and Yuancheng Yu 30:1–30:16

Linear-Time $(1+\varepsilon)$-Approximation Algorithms for Two-Line-Center Problems
Chaeyoon Chung, Anil Maheshwari, and Michiel Smid 31:1–31:17

Tensor Computation of Euler Characteristic Functions and Transforms
Jessi Cisewski-Kehe, Brittany Terese Fasy, Alexander McCleary, and Eli Quist ... 32:1–32:17

Near-Optimal Bounds for Parameterized Euclidean k-Means
Vincent Cohen-Addad, Karthik C. S., David Saulpic, and Chris Schwiegelshohn ... 33:1–33:17

Almost-Optimal Upper and Lower Bounds for Clustering in Low Dimensional Euclidean Spaces
Vincent Cohen-Addad, Karthik C. S., David Saulpic, and Chris Schwiegelshohn ... 34:1–34:16

On Computing the (Exact) Fréchet Distance with a Frog
Jacobus Conradi, Ivor van der Hoog, and Eva Rotenberg 35:1–35:20

Upward Book Embeddings of Partitioned Digraphs
Giordano Da Lozzo, Fabrizio Frati, and Ignaz Rutter 36:1–36:18

A Free Lunch: Manifolds of Positive Reach Can Be Smoothed Without Decreasing the Reach
Hana Dal Poz Kouřimská, André Lieutier, and Mathijs Wintraecken 37:1–37:19

On the Size of k-Irreducible Triangulations
Vincent Delecroix, Oscar Fontaine, and Arnaud de Mesmay 38:1–38:16

Locality Sensitive Hashing in Hyperbolic Space
Chengyuan Deng, Jie Gao, Kevin Lu, Feng Luo, and Cheng Xin 39:1–39:19

Computing the Intrinsic Delaunay Triangulation of a Closed Polyhedral Surface
Loïc Dubois 40:1–40:15

The Depth Poset Under Transpositions in the Filter
Herbert Edelsbrunner, Michał Lipiński, Marian Mrozek, Manuel Soriano-Trigueros, and Fedor Zimin 41:1–41:18

Compressed Data Structures for Heegaard Splitting
Henrique Ennes and Clément Maria 42:1–42:15

Unlabeled Multi-Robot Motion Planning with Improved Separation Trade-Offs
Tsuri Farhana, Omrit Filtser, and Shalev Goldshtein 43:1–43:16

Tilt Automata: Gathering Particles with Uniform External Control
Sándor P. Fekete, Jonas Friemel, Peter Kramer, Jan-Marc Reinhardt, Christian Rieck, and Christian Scheffer 44:1–44:19

Line Segment Visibility in Simple Polygons: Exact, Robust, Scalable Computation and Applications
Sándor P. Fekete, Prahlad Narasimhan Kasthurirangan, Phillip Keldenich, Fabian Kollhoff, Chek-Manh Loi, and Michael Perk 45:1–45:19

A Branch-And-Bound Algorithm for the Traveling Salesman Problem with Difficult Neighborhoods
Sándor P. Fekete, Rouven Kniep, Dominik Krupke, and Michael Perk 46:1–46:20

Computing the Skyscraper Invariant
Marc Fersztand and Jan Jendrysiak 47:1–47:23

FPT Approximations for Capacitated Sum of Radii and Diameters
Arnold Filtser and Ameet Gadekar 48:1–48:18

Algorithms for Euclidean Distance Matrix Completion: Exploiting Proximity to Triviality
Fedor V. Fomin, Petr A. Golovach, M. S. Ramanujan, and Saket Saurabh 49:1–49:15

Separators for Intersection Graphs of Spheres
Jacob Fox and Jonathan Tidor 50:1–50:16

First-Order Logic and Twin-Width for Some Geometric Graphs
Colin Geniet, Gunwoo Kim, and Lucas Meijer 51:1–51:16

Online Packing of Orthogonal Polygons
Tim Gerlach, Benjamin Hennies, and Linda Kleist 52:1–52:17

Constructing Doppelgängers of Greedy Geometric Spanners in Practice
Anirban Ghosh 53:1–53:21

Approximating Pareto Sum via Bounded Monotone Min-Plus Convolution
Geri Gokaj, Marvin Künnemann, Sabine Storandt, and Carina Truschel 54:1–54:21

Linear Time Single-Source Shortest Path Algorithms in Euclidean Graph Classes
Joachim Gudmundsson, Yuan Sha, and Sampson Wong 55:1–55:14

Optimal Randomized Clustering of Matrices
Mustafa Alper Gunes and Assaf Naor 56:1–56:18

Singular Arrange and Traverse Algorithm for Computing Reeb Spaces of Bivariate PL Maps
Petar Hristov, Ingrid Hotz, and Talha Bin Masood 57:1–57:17

On Sparse Representations of 3-Manifolds
Kristóf Huszár and Clément Maria 58:1–58:18

Improved Bound for the k-Variate Elekes–Rónyai Theorem
Yaara Jahn and Orit E. Raz 59:1–59:15

Space-Efficient Approximate Spherical Range Counting in High Dimensions
Andreas Kalavas and Ioannis Psarros 60:1–60:15

Complements of Finite Unions of Convex Sets
Chaya Keller and Micha A. Perles 61:1–61:18

Computing the Bottleneck Distance Between Persistent Homology Transforms
Michael Kerber and Elena Xinyi Wang 62:1–62:15

Unavoidable Patterns and Plane Paths in Dense Topological Graphs
Balázs Keszegh, Andrew Suk, Gábor Tardos, and Ji Zeng 63:1–63:15

Gap-ETH-Tight Algorithms for Hyperbolic TSP and Steiner Tree
Sándor Kisfaludi-Bak, Saeed Odak, Satyam Singh, and Geert van Wordragen 64:1–64:17

Near-Optimal Dynamic Steiner Spanners for Constant-Curvature Spaces
Sándor Kisfaludi-Bak and Geert van Wordragen 65:1–65:17

Fast Nearest Neighbor Search for ℓ_p Metrics
Robert Krauthgamer and Nir Petruschka 66:1–66:9

Non-Dissective Coverings by Planks
Andrey Kupavskii and János Pach 67:1–67:11

Optimal Bounds for Spanners and Tree Covers in Doubling Metrics
An La, Hung Le, Shay Solomon, Cuong Than, Vinayak, Shuang Yang, and Tianyi Zhang 68:1–68:16

Approximate Dynamic Nearest Neighbor Searching in a Polygonal Domain
Joost van der Laan, Frank Staals, and Lorenzo Theunissen 69:1–69:16

Tree-Like Shortcuttings of Trees
Hung Le, Lazar Milenković, Shay Solomon, and Cuong Than 70:1–70:15

Approximating Euclidean Shallow-Light Trees
Hung Le, Shay Solomon, Cuong Than, Csaba D. Tóth, and Tianyi Zhang 71:1–71:16

Topological Simplification Guided by Forbidden Regions
Jakub Leśkiewicz, Bartosz Furmanek, Michał Lipiński, and Dmitriy Morozov 72:1–72:17

The Complete 10-Tetrahedra Census of Orientable Cusped Hyperbolic 3-Manifolds
Shana Yunsheng Li 73:1–73:15

Manifolds of Positive Reach, Differentiability, Tangent Variation, and Attaining the Reach
André Lieutier and Mathijs Wintraecken 74:1–74:16

Geodesics of Length Less Than πR in a Set of Reach R Are Unique and Continuous with Respect to the Endpoints
André Lieutier and Mathijs Wintraecken 75:1–75:12

A Fast Algorithm for the Hecke Representation of the Braid Group, and Applications to the Computation of the HOMFLY-PT Polynomial and the Search for Interesting Braids
Clément Maria and Hoel Queffelec 76:1–76:18

Robust Algorithms for Path and Cycle Problems in Geometric Intersection Graphs
Malory Marin, Jean-Florent Raymond, and Rémi Watrigant 77:1–77:14

Mapping Chemical Space: Topological Data Analysis of Chemical Latent Space with Mapper
Dhruv Meduri, Chuan-Shen Hu, Cong Shen, Kelin Xia, and Bei Wang 78:1–78:20

D-GRIL: End-To-End Topological Learning with 2-Parameter Persistence
Soham Mukherjee, Shreyas N. Samaga, Cheng Xin, Steve Oudot, and Tamal K. Dey 79:1–79:17

Hardness of High-Dimensional Linear Classification
Alexander Munteanu, Simon Omlor, and Jeff M. Phillips 80:1–80:16

Optimal-Cost Construction of Shallow Cuttings for 3-D Dominance Ranges in the I/O-Model
Yakov Nekrich and Saladi Rahul 81:1–81:15

A Persistent Version of Latschev's Theorem
Steve Oudot and Lukas Waas 82:1–82:17

Erdős's Unit Distance Problem and Rigidity
János Pach, Orit E. Raz, and József Solymosi 83:1–83:9

The Voronoi Diagram of Four Lines in $\mathbb{R}^3$
Evanthia Papadopoulou and Zeyu Wang 84:1–84:17

ETH-Tight Complexity of Optimal Morse Matching on Bounded-Treewidth Complexes
Geevarghese Philip and Erlend Raa Vågset 85:1–85:19

Expansion of Trivariate Polynomials Using Proximity
Orit E. Raz 86:1–86:10

Robustness of Persistent Topological Features and Minimum Homological Cuts
Pepijn Roos Hoefgeest and Lucas Slot 87:1–87:15

Integer Points in Dilates of Polytopes
Shubhangi Saraf and Narmada Varadarajan 88:1–88:9

Approximating Convex Hulls via Range Queries
Thomas Schibler, Jie Xue, and Jiumu Zhu 89:1–89:15

Better Sampling Bounds for Restricted Delaunay Triangulations and a Star-Shaped Property for Restricted Voronoi Cells
Jonathan Richard Shewchuk 90:1–90:16

The Hierarchy of Manifolds in a Stratification of the Set of Equivalent Linear Neural Networks
Jonathan Richard Shewchuk and Sagnik Bhattacharya 91:1–91:19

Estimation of Conformal Metrics
Jérôme Taupin 92:1–92:15

Simplicial Approximation to CW Complexes with Spherical Delaunay Triangulations
Raphaël Tinarrage 93:1–93:22

Mixup Barcodes: Quantifying Geometric-Topological Interactions Between Point Clouds
Hubert Wagner, Nickolas Arustamyan, Matthew Wheeler, and Peter Bubenik 94:1–94:19

An Optimal Algorithm for Computing Many Faces in Line Arrangements
Haitao Wang 95:1–95:14

Media Expositions

Sliding Cubes in Parallel
Hugo A. Akitaya, Joseph Dorfer, Peter Kramer, Christian Rieck, Soham Samanta, Gabriel Shahrouzi, and Frederick Stock 96:1–96:5

"Visualizing" the CG Community
Oswin Aichholzer, Hugo A. Akitaya, Anna Brötzner, Peter Kramer, Christian Rieck, and Frederick Stock 97:1–97:4

Interactive Uniform Floodlight Illumination and Rotating Rays Voronoi Diagrams
Carlos Alegría, Ioannis Mantas, Marko Savić, and Martin Suderland 98:1–98:7

Proximity Alert: Ipelets for Neighborhood Graphs and Clustering
Gitan Balogh, June Cagan, Bea Fatima, Auguste H. Gezalyan, Danesh Sivakumar, Arushi Srinivasan, Yixuan Sun, Vahe Zaprosyan, and David M. Mount 99:1–99:8

Visualizing Higher Order Structures, Overlap Regions, and Clustering in the Hilbert Geometry
Hridhaan Banerjee, Soren Brown, June Cagan, Auguste H. Gezalyan, Megan Hunleth, Veena Kailad, Chaewoon Kyoung, Rowan Shigeno, Yasmine Tajeddin, Andrew Wagger, Kelin Zhu, and David M. Mount 100:1–100:6

From Chaos to Continents: Voronoi-Based Procedural Terrain Generation with Hydrology and 3D Visualization
Batsambuu Batbold and Lori Ziegelmeier 101:1–101:7

Tracking a Set of Moving Objects with Minimal Peak Power
Sándor P. Fekete, Malte Hoffmann, Chek-Manh Loi, and Michael Perk 102:1–102:7

Scalable Algorithmic Methods for Simulating Heavy-Rain Events
Sándor P. Fekete, Phillip Keldenich, Michael Perk, and Tobias Wallner 103:1–103:6

Interactive Visualization and Verification Tools for Tesseract Path Unfoldings
Soham Samanta, Hugo A. Akitaya, Erik Demaine, and Martin Demaine 104:1–104:5

CG Challenges

ETH Flippers Approach to Parallel Reconfiguration of Triangulations: SAT Formulation and Heuristics
Lorenzo Battini and Marko Milenković 105:1–105:6

Engineering Greedy Heuristics and Simulated Annealing Methods for the Median Triangulation Under the Parallel Flip Distance
Jacobus Conradi, Benedikt Kolbe, Philip Mayer, Jonas Sauer, and Jack Spalding-Jamieson 106:1–106:7

Shadoks Approach to Parallel Reconfiguration of Triangulations
Guilherme D. da Fonseca, Fabien Feschet, and Yan Gerard 107:1–107:7

CG#Hunters Approach to Central Triangulation Under Parallel Flip Operations
Jaegun Lee, Seokyun Kang, Hyeonseok Lee, Hyeyun Yang, and Taehoon Ahn 108:1–108:8

Preface

The 42nd International Symposium on Computational Geometry (SoCG 2026) was held in New Brunswick, NJ, USA, June 2–5, 2026, as part of the Computational Geometry Week (CG Week 2026).

To strengthen and promote research on practical aspects of computational geometry and topology, as well as connections to other fields, SoCG 2026 introduced a new track that specifically focuses on these aspects, such as implementation and software, engineering, experimentation, and applications. Thus, for the first time in 25 years, the submission and review process was organized in two tracks: Track P (Practice) and Track T (Theory). Authors could also specify that their submission falls in between the tracks if it makes relevant contributions of both theoretical and practical nature.

The symposium received a record number of 272 submissions, of which 217 were for Track T, 40 in between, 14 for Track P, and one was withdrawn later. After a thorough review process, in which each paper was evaluated by several (typically three or four) independent reviewers, the program committee accepted 95 papers for presentation, of which 76 were from Track T, 13 in between, and 6 from Track P. These proceedings contain extended abstracts of the accepted papers, limited to 500 lines (excluding references). If any supporting material does not fit in the line limit, it is persistently available at a public repository and referenced in the corresponding extended abstract.

The Best Theory Paper Award of SoCG 2026 went to the paper "On the size of k-irreducible triangulations" by Vincent Delecroix, Oscar Fontaine and Arnaud de Mesmay. The Best Practice Paper Award of SoCG 2026 went to the paper "Constructing Doppelgängers of Greedy Geometric Spanners in Practice" by Anirban Ghosh. The Best Student Paper Award of SoCG 2026 went to the paper "Fréchet Distance in the Imbalanced Case" by Lotte Blank. The Best Student Presentation Award was determined and announced at the symposium, based on ballots cast by the attendees. A selection of papers was invited to submit extended versions to forthcoming special issues of Discrete & Computational Geometry (DCG) and the Journal of Computational Geometry (JoCG) dedicated to the symposium.

The SoCG Test of Time Award 2026 went to "An efficient algorithm for planning collision-free translational motion of a convex polygonal object in 2-dimensional space amidst polygonal obstacles" by Klara Kedem and Micha Sharir, published in SoCG 1985, and to "Locality-sensitive hashing scheme based on p-stable distributions" by Mayur Datar, Nicole Immorlica, Piotr Indyk and Vahab S. Mirrokni, published in SoCG 2004.

The scientific program of CG Week 2026 was enriched by two distinguished invited speakers. An invited talk "From shape comes meaning: Bringing geometric intuition to the ethics of AI" was given by Suresh Venkatasubramanian from Brown University. A second invited talk "New directions in sublinear time algorithms" was delivered by Ronitt Rubinfeld from the Massachusetts Institute of Technology. We thank the plenary speakers for kindly accepting our invitation.

In addition to the technical papers, there were nine submissions to the multimedia exposition. Submissions were reviewed and all of them were accepted for presentation. The extended abstracts that describe these submissions are included in this proceedings volume. The multimedia content can be found at `https://computational-geometry.org/cg-week/media-exposition/2026/`.

42nd International Symposium on Computational Geometry (SoCG 2026).
Editors: Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



The 8th Computational Geometry Challenge was part of CG Week 2026. The challenge problem was about finding a central triangulation under parallel flip operations. This year, eighteen teams participated in the challenge. These proceedings contain contributions by the four top-placed teams describing their winning approaches.

We thank the authors of all submitted contributions. We are most grateful to the members of the SoCG Program Committee, the Media Exposition (CG:ME) Committee, and the CG Challenge Committee for their dedication, expertise, and hard work that ensured the high quality of the works in these proceedings. We are grateful for the assistance provided by the hundreds of external reviewers; without their help, it would have been impossible to run the selection process. Last but not least, we thank Philipp Kindermann for his meticulous work as the Proceedings Chair.

Many other people contributed to the success of SoCG 2026 and the entire CG Week. We are very grateful to the local organizers for their work in organizing the event. Finally, we thank all the members of the Test of Time Award Committee, the Young Researchers Forum (CG:YRF) Committee, the CG Challenge Advisory Board, the CG Society Board, the CG Society IT Committee, and the CG Steering Committee.

Hee-Kap Ahn
POSTECH

Michael Hoffmann
ETH Zürich

Amir Nayyeri
Oregon State University

Conference Organization

SoCG Program Committee

Track P

- Chao Chen (Stony Brook University)
- Otfried Cheong (Scalgo)
- Andreas Fabri (GeometryFactory)
- Marc Glisse (Inria Saclay)
- John Hershberger (Siemens EDA)
- Michael Hoffmann (co-chair; ETH Zürich)
- Phillip Keldenich (TU Braunschweig)
- Dmitriy Morozov (Lawrence Berkeley National Laboratory)
- Sabine Storandt (University of Konstanz)
- Hubert Wagner (University of Florida)
- Jules Wulms (TU Eindhoven)

Track T

- Eyal Ackerman (University of Haifa at Oranim)
- Hee-Kap Ahn (co-chair; POSTECH)
- Pankaj Agarwal (Duke University)
- Martin Balko (Charles University)
- Huck Bennett (University of Colorado Boulder)
- Sarita de Berg (IT University of Copenhagen)
- Therese Biedl (University of Waterloo)
- Kevin Buchin (TU Dortmund)
- Sergio Cabello (University of Ljubljana)
- Jean Cardinal (Université Libre de Bruxelles)
- Mathieu Carrière (Inria Côte d'Azur)
- Éric Colin de Verdière (CNRS, LIGM, Marne-la-Vallée)
- Anne Driemel (Bonn University)
- Herbert Edelsbrunner (IST Austria)
- Arnold Filtser (Bar Ilan University)
- Xavier Goaoc (Université de Lorraine)
- Kristóf Huszár (TU Graz)
- Woojin Kim (KAIST)
- Sándor Kisfaludi-Bak (Aalto University)
- Hung Le (UMass Amherst)
- Amir Nayyeri (co-chair; Oregon State University)
- Yakov Nekrich (Michigan Technological University)
- Sharath Raghvendra (NC State)
- Paweł Rzążewski (Warsaw University of Technology)
- Frank Staals (Utrecht University)
- Primož Škraba (Queen Mary University of London)
- Shin-ichi Tanigawa (University of Tokyo)
- Katharine Turner (Australian National University)

- Carola Wenk (Tulane University)
- Sampson Wong (University of Copenhagen)
- Meirav Zehavi (Ben Gurion Univeristy)

SoCG Proceedings Chair

- Philipp Kindermann (Universität Trier)

CG Media Exposition (CG:ME) Committee

- Hugo Akitaya (University of Massachusetts Lowell)
- Maarten Löffler (Utrecht University)
- Greg Malen (Skidmore College)
- Joseph O'Rourke (co-chair; Smith College)
- Ryuhei Uehara (JAIST)
- Elena Wang (University of Fribourg)
- Alexandra Wesolek (University Potsdam)
- Lori Ziegelmeier (co-chair; Macalester College)

CG Challenge (CG:SHOP) Committee

- Oswin Aichholzer (TU Graz)
- Joseph Dorfer (TU Graz)
- Sándor Fekete (TU Braunschweig)
- Phillip Keldenich (TU Braunschweig)
- Peter Kramer (TU Braunschweig)
- Dominik Krupke (TU Braunschweig)
- Stefan Schirra (University of Magdeburg)

CG Challenge Advisory Board

- Bill Cook (University of Waterloo)
- Andreas Fabri (Geometry Factory)
- Dan Halperin (Tel Aviv University)
- Michael Kerber (TU Graz)
- Philipp Kindermann (Universität Trier)
- Joe Mitchell (Stony Brook University)
- Kevin Verbeek (TU Eindhoven)

SoCG Test of Time Award Committee

- Pankaj K. Agarwal (Duke University)
- Mark de Berg (TU Eindhoven)
- Siu-Wing Cheng (Hong Kong University of Science and Technology)
- Yusu Wang (University of California, San Diego)

Young Researchers Forum (CG:YRF) Program Committee

- André van Renssen (The University of Sydney)
- Carolin Rehs (Dortmund University)
- Evanthia Papadopoulou (Università della Svizzera italiana)
- Florian Russold (TU Graz)
- Håvard Bakke Bjerkevik (University at Albany)
- Jack Stade (University of Copenhagen)
- Meirav Zehavi (Ben-Gurion University)
- Peyman Afshani (chair) (Aarhus University)
- Rahul Saladi (Indian Institute of Science Bangalore)

Local Organizing Committee

- Jie Gao (Computer Science, Rutgers)
- David Pennock (DIMACS, Rutgers)
- Tamra Carpenter (DIMACS, Rutgers)

CG Society Board

- Benjamin Burton (University of Queensland)
- David Eppstein (University of California, Irvine)
- Bettina Speckmann (TU Eindhoven)
- Frank Staals (Utrecht University)

CG Society IT Committee

- Yossi Bokor Bleile (Aalborg University)
- Fabian Klute (Universitat Politècnica de Catalunya)
- Willem Sonke (TU Eindhoven)

CG Steering Committee

- Maike Buchin (Ruhr-Universität Bochum)
- Erin Chambers (University of Notre Dame)
- Anne Driemel (Universität Bonn)
- Marc van Kreveld (Utrecht University)
- Joe Mitchell (Stony Brook University)
- Wolfgang Mulzer (Freie Universität Berlin)

Additional Reviewers

- Eddie Aamari
- Fateme Abbasi
- Mikkel Abrahamsen
- Bernardo Ábrego
- Henry Adams
- Peyman Afshani
- Oswin Aichholzer
- Yariv Aizenbud
- Hugo Akitaya
- Shannon Algar
- Ángel Javier Alonso
- Enrique G. Alvarado
- Gergely Ambrus
- Shinwoo An
- Alexandr Andoni
- Sebastian Angrick
- Todor Antić
- Toshitaka Aoki
- Boris Aronov
- Sunil Arya
- Stav Ashur
- Dominique Attali
- Amir Azarmehr
- Alexandra Bac
- Sang Won Bae
- Sayan Bandyapadhyay
- Aritra Banik
- Luis Barba
- Saugata Basu
- Ulrich Bauer
- Daniel Beďatš
- Michael Bekos
- Matthias Bentert
- Sergey Bereg
- Lorenzo Beretta
- Mark de Berg
- Daniel Irving Bernstein
- Sujoy Bhore
- Ahmad Biniaz
- Mitchell Black
- Lotte Blank
- Jarosław Błasiok
- Thomas Bläsius
- Václav Blažej
- Omer Bobrowski
- Yossi Bokor Bleile
- Édouard Bonnet
- Prosenjit Bose
- Magnus Botnan
- Karl Bringmann
- Anna Brötzner
- Guillaume Brouillette
- Reilly Browne
- Morten Brun
- Peter Bubenik
- Moritz Buchem
- Maike Buchin
- Denys Bulavka
- Rhuaidi Antonio Burke
- Benjamin Burton
- Jonathan Bush
- Clément Canonne
- Dario Giuliano Cavallaro
- Eliška Červenková
- Dibyayan Chakraborty
- Parinya Chalermsook
- Erin Chambers
- T H Hubert Chan
- Timothy Chan
- L. Sunil Chandran
- Hsien-Chih Chang
- Laetitia Chapel
- Anamay Chaturvedi
- Samantha Chen
- Siu-Wing Cheng
- Oliver A. Chubet
- David Cohen-Steiner
- Jacobus Conradi
- Jonathan Conroy
- Jack Cordes
- Jean Cousty
- James Cruickshank
- Monika Csikos
- Justin Curry
- Radu Curticapean
- Suryendu Dalal
- Rathish Das
- Syamantak Das

42nd International Symposium on Computational Geometry (SoCG 2026).
Editors: Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- Minati De
- Riley Decker
- Vincent Despré
- Sean Dewar
- Tamal K. Dey
- Hu Ding
- Michael Dinitz
- Michael Gene Dobbins
- Ondřej Draganov
- Lukas Drexler
- Ran Duan
- Loïc Dubois
- Vida Dujmovic
- Zeev Dvir
- Nadav Dym
- Eduard Eiben
- David Eppstein
- Jeff Erickson
- Esther Ezra
- Rolf Fagerberg
- Mohammad Farshi
- Stefan Felsner
- Marek Filakovský
- Christopher Fillmore
- Omrit Filtser
- Orr Fischer
- Emily Fox
- Mathew Francis
- Nóra Frankl
- Fabrizio Frati
- Florian Frick
- Ulderico Fugacci
- Kaito Fujii
- Niloufar Fuladi
- Stefan Funke
- Koki Furukawa
- Ameet Gadekar
- Harmender Gahlawat
- Ian Gallagher
- Robert Ganian
- Jie Gao
- Alexey Garber
- Bernd Gärtner
- Ellen Gasparovic
- Cameron Gates Rudd
- Arijit Ghosh
- Panos Giannopoulos
- Matt Gibson-Lopez
- Barbara Giunti
- Erwin Glazenburg
- Jelena Glisic
- Geri Gokaj
- Petr A. Golovach
- Alexander Golovnev
- Timothy Gomez
- Daniel Gonçalves
- Lee-Ad Gottlieb
- Dejan Govc
- Attila Gyulassy
- Dan Halperin
- Thekla Hamm
- Jakob Hansen
- Sariel Har-Peled
- Herman Haverkort
- Alexander He
- Meng He
- Peter Heiss Synak
- Stephan Held
- Gregory Henselman-Petrusek
- Adam Hesterberg
- Robert Hickingbotham
- Petr Hliněný
- Benjamin Holmgren
- Andreas F. Holmsen
- Ivor van der Hoog
- Gary Hoppenworth
- Thijs van der Horst
- Erfan Hosseini Sereshgi
- Tao Hou
- Haoqiang Huang
- Qijia Huang
- Zhengcheng Huang
- Stefan Huber
- Tanmay Inamdar
- Ragesh Jaiswal
- Vít Jelínek
- Jan Jendrysiak
- Mark Jerrum
- Zhihan Jin
- Howy Jordan
- Gwenaël Joret
- Antonia Kalb
- Iyad Kanj
- Jarkko Kari
- Karthik C. S.
- J. Mark Keil

- Chaya Keller
- Michael Kerber
- Balázs Keszegh
- Arindam Khan
- Donghan Kim
- Jisu Kim
- Philipp Kindermann
- Katharina Klost
- Fabian Klute
- Kristin Knorr
- Tsuyoshi Kobayashi
- Yasuaki Kobayashi
- Yusuke Kobayashi
- Florent Koechlin
- Noleen Koehler
- Benedikt Kolbe
- Jochen Konemann
- Matias Korman
- Ramin Kosfeld
- Peter Kramer
- Robert Krauthgamer
- Tomasz Krawczyk
- Marc van Kreveld
- Sascha Kurz
- Jan Kynčl
- An La
- Théo Lacombe
- Nathaniel Lahn
- Claudia Landi
- Sylvain Lazard
- Francis Lazarus
- Janis Lazovskis
- Vadim Lebovici
- Wonjun Lee
- Fabian Lenzen
- Michael Lesnick
- Jingyi Li
- Yuhao Li
- Joan Licata
- André Lieutier
- Sunhyuk Lim
- Uzu Lim
- Michał Lipiński
- Enhao Liu
- Solomon Lo
- William Lochet
- Maarten Löffler
- Georg Loho
- David Loiseaux
- Alexandre Louvet
- Corentin Lunel
- Haoran Luo
- Yuan Luo
- Sepideh Mahabadi
- Sushovan Majhi
- Sucharita Mallick
- Clement Maria
- Lucas Meijer
- Facundo Memoli
- Arnaud de Mesmay
- Wouter Meulemans
- Frédéric Meunier
- Bertrand Michel
- Lazar Milenković
- Till Miltzow
- Majid Mirzanezhad
- Konstantin Mischaikow
- Tobias Mömke
- Morteza Monemizadeh
- Anthea Monod
- Guillaume Moroz
- Francis Motta
- David Mount
- Jonathan Mousley
- Wolfgang Mulzer
- Andrea Munaro
- Liz Munch
- Nabil Mustafa
- Atsuhiro Nakamoto
- Tom Needham
- Frank Nielsen
- Arnur Nigmetov
- Bengt Nilsson
- André Nusser
- Ryoshun Oba
- Saeed Odak
- Naoto Ohsaka
- Adam Onus
- Tim Ophelders
- Joachim Orthaber
- George Osipov
- Steve Oudot
- Ziyad Oulhaj
- Dömötör Pálvölgyi
- Evanthia Papadopoulou
- Pedro Paredes

- Salman Parsa
- Zahra Parsaeian
- Mano Prakash Parthasarathi
- Pavel Paták
- Matteo Pegoraro
- Sarah Percival
- Michael Perk
- Jonathan Perry
- Tom Peters
- Tuyen Pham
- Jeff Phillips
- Théo Pierron
- Lukas Plätz
- Kacper Pluta
- Valentin Polishchuk
- Marc Pouget
- Lionel Pournin
- Siddharth Pritam
- Ioannis Psarros
- Sergey Pupyrev
- Demi Qin
- Eli Quist
- Benjamin Raichel
- Rajiv Raman
- Yingli Ran
- Malin Rau
- Mathias Rav
- Dror Rawitz
- Rahul Raychaudhury
- Carolin Rehs
- Emily Ribando-Gros
- Marena Richter
- Christian Rieck
- Oliver Roche-Newton
- Günter Rote
- Tristan Roussillon
- Mael Rouxel-Labbé
- Martin Royer
- Natan Rubin
- Florian Russold
- Daniel Rutschmann
- Arsenii Sagdeev
- Rahul Saladi
- Eric Samperton
- Francisco Santos Leal
- Shay Sapir
- Ramprasad Saptharishi
- Maria Saumell
- Torben Scheele
- Patrick Schnider
- Robert Schweller
- Chris Schwiegelshohn
- Martin P. Seybold
- Yuan Sha
- Donald R. Sheehy
- Jonathan Shewchuk
- Pouyan Shirzadian
- Vin de Silva
- Rodrigo I. Silveira
- Francesco Silvestri
- Sandeep Silwal
- Kirill Simonov
- Karamjeet Singh
- Satyam Singh
- Michiel Smid
- Gleb Smirnov
- Shakhar Smorodinsky
- József Solymosi
- Willem Sonke
- Jan Soukup
- Jack Spalding-Jamieson
- Jonathan Spreer
- Raghavendra Sridharamurthy
- Jack Stade
- Fabian Stehn
- Raphael Steiner
- Noah Stephens-Davidowitz
- Elizaveta Streltsova
- Aditya Subramanian
- Akira Suzuki
- Yusuke Suzuki
- Krisztina Szilagyi
- Kiyoshi Takeuchi Romo
- Xuehou Tan
- Martin Tancer
- Yufei Tao
- Javier Tejel
- Cuong Than
- Leonidas Theocharous
- Amber Thrall
- Anastasiia Tkachenko
- Josef Tkadlec
- Francesca Tombari
- István Tomon
- Yiying Tong
- Csaba D. Tóth

- Konstantinos Tsakalidis
- Eliad Tsfadia
- Torsten Ueckerdt
- Ryuhei Uehara
- Léo Valque
- Kasturi Varadarajan
- Santosh Vempala
- Antoine Vigneron
- Vinayak Vinayak
- Ziga Virk
- Thuy-Duong Vuong
- Mirko H. Wagner
- Uli Wagner
- Bartosz Walczak
- Elena Wang
- Fan Wang
- Haitao Wang
- Qingsong Wang
- Yusu Wang
- Evelyn Warton
- Alexandra Weinberger
- Andreas Wiese
- Mathijs Wintraecken
- Arne Wolf
- Geert van Wordragen
- Lasse Wulf
- Tim Wylie
- Cheng Xin
- Jie Xue
- Chao Yan
- Jungwoo Yang
- Shuang Yang
- Jiachen Yao
- Keegan Yao
- Ka Man (Ambrose) Yim
- Iris Yoon
- Sang Duk Yoon
- Yelena Yuditsky
- Hanwen Zhang
- Wanchen Zhao
- Yiming Zhao
- Da Wei Zheng
- Wan Zhengchao
- Samson Zhou
- Johannes Zink
- Stanislav Živný

Covering and Partitioning Complex Objects with Small Pieces

Anders Aamand
Department of Computer Science, University of Copenhagen, Denmark

Mikkel Abrahamsen
Department of Computer Science, University of Copenhagen, Denmark

Reilly Browne
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Mayank Goswami
Department of Computer Science, Queens College CUNY, New York, NY, USA

Prahlad Narasimhan Kasthurirangan
Department of Applied Mathematics and Statistics, Stony Brook University, NY, USA

Linda Kleist
Department of Informatics, Universtity Hamburg, Germany

Joseph S. B. Mitchell
Department of Applied Mathematics and Statistics, Stony Brook University, NY, USA

Valentin Polishchuk
Communications and Transport Systems, Linköping Univeristy, Sweden

Jack Stade
Department of Computer Science, University of Copenhagen, Denmark

Abstract

We study the problems of covering or partitioning a polygon P (possibly with holes) using a minimum number of *small pieces*, where a small piece is a connected sub-polygon contained in an axis-aligned unit square. For covering, we seek to write P as a union of small pieces, and in partitioning, we furthermore require the pieces to be pairwise interior-disjoint. We show that these problems are in fact equivalent: Optimum covers and partitions have the same number of pieces.

For covering, a natural local search algorithm repeatedly attempts to replace k pieces from a candidate cover with $k-1$ pieces. In two dimensions and for sufficiently large k, we show that when no such swap is possible, the cover is a $1+\mathcal{O}(1/\sqrt{k})$ approximation, hence obtaining the first PTAS for the problem. Prior to our work, the only known algorithm was a 13-approximation that only works for polygons without holes [Abrahamsen and Rasmussen, SODA 2025]. In contrast, in the three dimensional version of the problem, for a polyhedron P of complexity n, we show that it is NP-hard to approximate an optimal cover or partition to within a factor that is logarithmic in n, even if P is simple, i.e., has genus 0 and no holes.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Packing and covering problems; Theory of computation → Approximation algorithms analysis

Keywords and phrases Covering, partitioning, polygon, small piece, PTAS

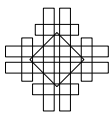
Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.1

Related Version *Full Version*: http://arxiv.org/abs/2603.23216 [1]

Funding *Anders Aamand*: This work was supported by the VILLUM Foundation grant 54451.
Mikkel Abrahamsen: Supported by Independent Research Fund Denmark, grant 1054-00032B, and by the Carlsberg Foundation, grant CF24-1929.
Mayank Goswami: Supported by National Science Foundation CCF-2503086.



© Anders Aamand, Mikkel Abrahamsen, Reilly Browne, Mayank Goswami, Prahlad Narasimhan Kasthurirangan, Linda Kleist, Joseph S. B. Mitchell, Valentin Polishchuk, and Jack Stade;
licensed under Creative Commons License CC-BY 4.0



42nd International Symposium on Computational Geometry (SoCG 2026).
Editors: Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri; Article No. 1; pp. 1:1–1:16



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Prahlad Narasimhan Kasthurirangan: Some of this work was carried out during a research stay at TU Braunschweig. Partially supported by the US Office of Naval Research under grant no. N00014-26-1-2088.
Jack Stade: Supported by Independent Research Fund Denmark, grant 1054-00032B, and by the Carlsberg Foundation, grant CF24-1929.

1 Introduction

Partitioning large objects into pieces of bounded size is a fundamental problem. One practical motivation comes from 3D printing and related manufacturing technologies: we can only produce objects that fit within the printing volume, which can be assumed to be a unit cube. If the goal is to produce a large part P, we must partition P into smaller pieces, $Q_1, \ldots, Q_k$, each of which fits within the unit cube. The small pieces can then be assembled to form the desired part P. A natural objective is to minimize the number, k, of pieces. Heuristics have been proposed to solve this problem [31, 18, 26, 30, 38, 36].

Similar partitioning problems arise in everyday settings. For instance, one might disassemble an entire broccoli head, the peeled stem included, into pieces suitable for consumption with minimal indignity. Yet even in this humble task, one strives to avoid the disarray of producing an excessive multitude of fragments. Recently, Abrahamsen and Rasmussen [5] initiated a theoretical study of partitioning polygons into few small pieces. They provided constant-factor approximation algorithms when the domain P is a simple polygon (no holes), for various definitions of what it means for a piece Q_i to be "small," each with its own algorithm and approximation factor. A particularly simple version demands that each piece Q_i be connected and contained in an axis-aligned unit square. Abrahamsen and Stade [6] showed that this version is NP-hard even when P is a simple polygon.

In this paper, we consider the problems of partitioning and covering objects with small pieces in two and three dimensions. We consider the setting where a small piece is defined as a connected subset of P contained in an axis-aligned unit square (in two dimensions) or unit cube (in three dimensions). We obtain the following results.

- We present a PTAS for covering a polygon (possibly with holes) into a minimum number of small pieces (Section 2).
- For polygons (possibly with holes), we show that the small-cover and small-partition problem have the same optimal value (Section 2.4).
- We prove that in 3D, covering or partitioning a polyhedron P of description complexity n with small pieces is NP-hard to approximate within a logarithmic factor of n, even when P is simply-connected and contained in a box of size $1 \times 2 \times 3$ (Section 3).

Our PTAS works by local search: we start with a candidate small cover and then repeatedly try to find a set of k pieces that can be swapped for a set of $k-1$ pieces. We show that this gives a $1 + \mathcal{O}(1/\sqrt{k})$ approximation algorithm, when k is sufficiently large. A result by Roy, Govindarajan, Raman, and Ray [34] shows that local search is a PTAS for discrete Set-Cover when the candidate pieces satisfy a certain *non-piercing* property. Two simply-connected pieces P, Q are *non-piercing* if $P \setminus Q$ and $Q \setminus P$ are connected. We show that a set of maximal small pieces satisfies a slightly weaker non-piercing property, but we are able to modify the pieces slightly so that the result from [34] can be used. We are then able to analyze our "continuous" local search algorithm using this result–unlike the discrete case studied in [34], note that our search space is infinite.

For many optimization algorithms, local search can be used to obtain a PTAS when a certain related graph has sublinear separators (see [8] for details). We highlight the fact that this is non-obvious for the small-cover problem. Local search gives a PTAS for *packing*

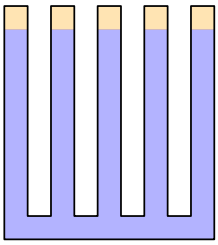


Figure 1 A comb-shaped polygon might require an arbitrarily large number of pieces for a small cover, even if it fits in a box of size $1 \times (1 + \varepsilon)$.

axis-aligned unit squares into a polyhedron in any constant dimension, since a density argument can be used to obtain the required separation bounds (see for example [24]). In contrast, for small covers, we insist that the pieces be connected, which means that there are "comb-shaped" polygons requiring many pieces, even though the polygon itself almost fits in a unit square; see Figure 1. In 3D, we show that it is NP-hard to approximate optimal small cover or partition to within a factor of $(1 - \alpha)\ln(n)$, for any $\alpha > 0$, where n is the number of bits needed to describe the input polyhedron. We also observe (Theorem 10) that for small pieces, covering and partitioning are equivalent, in the sense that the smallest number of pieces in a cover and in a partition is the same. This is in stark contrast with other geometric settings where not only the optima differ, but also covering seems considerably harder than partitioning. For instance, it is $\exists\mathbb{R}$-complete to cover a simple polygon with the minimum number of star polygons [3] (that is, the art-gallery problem) or convex polygons [2], while the corresponding partitioning problems are solvable in polynomial time [4, 16]. Details of all proofs can be found in the full version [1].

1.1 Related Work

The small cover and partitioning problems belong to the class of *decomposition problems*, which form a large, established subfield in computational geometry. In all of these problems, we want to *decompose* a polygon P into connected polygonal *pieces* that must respect certain restrictions. Here, the union of the pieces must be P, and we usually seek a decomposition into as few pieces as possible, in which case the decomposition is called *optimal*. A decomposition that allows the pieces to overlap is called a *cover*; if thee pieces are pairwise interior-disjoint, it is called a *partition*. Depending on the assumptions about the input polygon P and the requirements on the pieces, this leads to a wealth of interesting problems. There is a vast literature about such decomposition problems, as documented in several books and survey papers that give an overview of the state-of-the-art at their respective times of publication [35, 17, 15, 28, 29, 32, 33].

Earlier works studied variants of the small partitioning problem without Steiner points [20, 13], i.e., each vertex of a piece Q_i is a vertex of P. As an undesirable consequence such a partition often does not exist, e.g., if an edge of the polygon P is too long to fit in a piece.

Motivated by indoor localization using lasers, Arkin, Das, Gao, Goswami, Mitchell, Polishchuk, and Tóth [7] studied partitioning a polygon with chords (maximal segments within P), which may model laser beams that serve as "tripwires"; the goal is to cut the polygon into bounded pieces using the fewest lasers. Motivated by communication graphs for mobile robots, Fekete, Kamphans, Kröller, Mitchell, and Schmidt [23] (see also the video [9]) showed how to compute a Steiner triangulation of a simple polygon, whose edge lengths are bounded; they presented a 3-approximation for minimizing the number of Steiner points.

Worman [37] showed NP-hardness of partitioning a polygon with holes into a minimum number of pieces contained in axis-aligned unit squares, without Steiner points. More recently, Buchin and Selbach [13] proved NP-hardness of partitioning and covering with fat pieces. Combinatorial aspects of partitioning into small[er] pieces are classical problems in mathematics. For instance, *Borsuk's problem* is to partition a unit-diameter convex body (in any dimension) into fewest pieces of diameter strictly less than 1; see [12, 25, 27] for the rich history of the problem. *Conway's fried potato problem* [10, 11, 14, 19] deals with minimizing the in-radius of the pieces obtained by a specified number of hyperplane cuts of a given convex polyhedron (in d dimensions); in our terminology, Conway's problem is a *dual* small partitioning problem.

1.2 Problem Formulation

We say that a polygon Q (possibly with holes) is *small* if Q is contained in an axis-aligned unit square. We often call small polygons small *pieces*. A set $\mathcal{Q}$ of small pieces is a *small cover* of a polygon P if $\bigcup \mathcal{Q} = P$; note in particular that $Q \subseteq P$ for each small piece $Q \in \mathcal{Q}$. We say that a piece $Q \in \mathcal{Q}$ is *complete* if Q is the closure of a connected component of the intersection $P^\circ \cap S$, where P° is the interior of P and S is an axis-aligned unit square. A simpler definition would be that Q is complete if it is a connected component of $P \cap S$, but that could be a degenerate polygon, which we by definition don't allow. A small cover $\mathcal{Q}$ of P is a *small partition* of P if furthermore the pieces in $\mathcal{Q}$ are pairwise interior-disjoint. In this paper, we study the problems of finding small covers and small partitions consisting of a minimum number of pieces for a given polygon P, which may have holes. We denote these problems Small-Cover and Small-Partition, respectively.

2 A PTAS for Small-Cover in 2 dimensions

In this section, we present an $(1+\varepsilon)$-approximation algorithm for Small-Cover when the input is a polygon P, possibly with holes. In Section 2.1, we establish tools that allow us to use local search. We then show in Section 2.2 how to use these techniques to handle polygons which intersect at most D squares in the integer grid. The running time of our algorithm are polynomial in D. Finally, in Section 2.3, we extend our ideas to handle polygons that might have exponentially large diameter. We note that, for the purposes of constructing approximate solutions for Small-Cover, we will only consider *complete* small pieces, since any small piece that fits in a square S must cover a subset of the complete piece covering the same component of $P^\circ \cap S$.

2.1 Fundamentals for Local Search

Roy, Govindarajan, Raman, and Ray [34] studied the geometric set cover problem with so-called non-piercing regions. Two compact simply-connected regions A and B are *non-piercing* if $A \setminus B$ and $B \setminus A$ are both connected. We say that A and B intersect *properly* if the intersection of the boundaries $\partial A \cap \partial B$ consists of a finite set of points and in each of these points, the boundaries cross, i.e., tangential intersections are not allowed. We say that a set of compact, simply-connected regions is *properly non-piercing* if the regions are pairwise non-piercing and intersect properly.

▶ **Lemma 1** ([34]). *For a universal constant C the following holds: Let X be a (finite) set of points in $\mathbb{R}^2$ and suppose that $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$ and $\mathcal{R} = \{R_1, \ldots, R_r\}$ are sets of connected regions such that $\mathcal{Q}$ and $\mathcal{R}$ each cover X. If the regions in $\mathcal{Q} \cup \mathcal{R}$ are properly non-piercing and $r \geq (1+\varepsilon)q$ for some $1 \geq \varepsilon > 0$, then there is some set of at most C/ε^2 of the pieces in $\mathcal{R}$ that can be replaced by a strictly smaller set of pieces in $\mathcal{Q}$ to obtain a cover of X.*

We start by adapting this result to the setting of covering a polygon rather than a discrete set of points. We show the following:

▶ **Lemma 2.** *There is a constant C such that the following holds: Let P be a polygon (possibly with holes) and let $\mathcal{Q}$ be an optimal small cover of P consisting of q complete pieces, and let $\mathcal{R}$ be another small cover of P consisting of r complete pieces. Then for any $1 \geq \varepsilon > 0$, if $r \geq (1+\varepsilon)q$, then there is a set of at most C/ε^2 of the pieces in $\mathcal{R}$ that can be replaced by a strictly smaller set of pieces in $\mathcal{Q}$ to obtain a small cover of P.*

It is relatively straightforward to prove Lemma 2 given Lemma 1. The strategy is to put a point in the interior of each cell of the overlay of $\mathcal{Q}$ and $\mathcal{R}$. A subset of $\mathcal{Q} \cup \mathcal{R}$ covers P if and only if it covers each of these points. The pieces in $\mathcal{Q} \cup \mathcal{R}$ are almost non-piercing, but need to be modified slightly in order to remove degeneracies and holes. The process is demonstrated informally in Figure 2 and uses an idea from [21]. The following lemma says that these modifications exist:

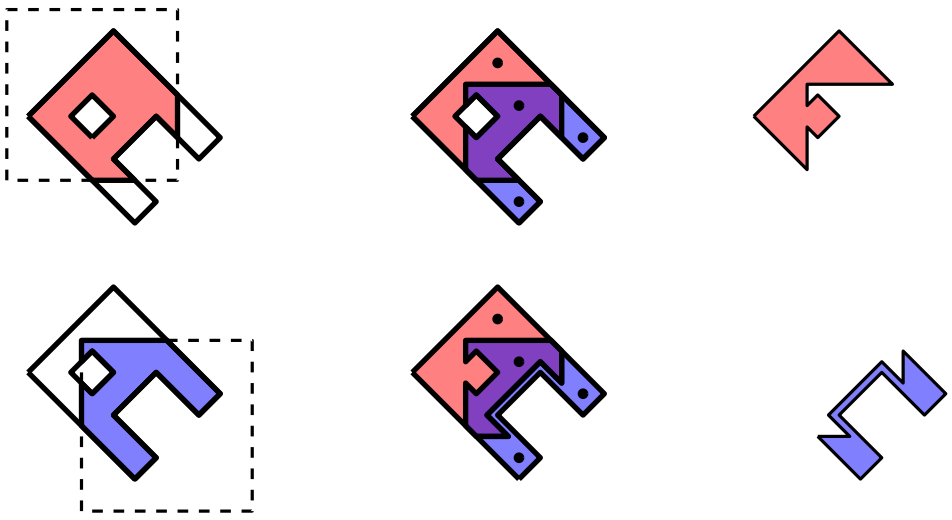


Figure 2 Given a finite set of complete small pieces that cover P, we obtain an equivalent instance of non-piercing geometric set cover. Left: a polygon and two complete small pieces. Top middle: the polygon and pieces cut the polygon into 4 cells, and we place a point in each cell. These pieces are piercing, since removing the red piece from the blue piece would disconnect it. Additionally, the red piece has a hole. Bottom middle: modifying the pieces to remove the hole and make them non-piercing. The hole in the red piece is filled, and an interval of the boundary is offset inwards so as to not pierce the blue piece. Right: removing the modified blue piece from the red piece (top) or the modified red piece from the blue piece (bottom). The remaining parts are connected, so the modified pieces are non-piercing.

▶ **Lemma 3.** *Consider a polygon P and a small cover $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$ of P consisting of complete small pieces. Consider the overlay of the pieces in $\mathcal{Q}$, choose a point in the interior of each region contained in P, and let X be the set of these points. There exists a set $\mathcal{Q}' = \{Q'_1, \ldots, Q'_q\}$ of simply-connected pieces that are properly non-piercing such that $Q_i \cap X = Q'_i \cap X$ for all $i \in \{1, \ldots, q\}$. It holds in particular that for any $I \subset \{1, \ldots, q\}$, we have $X \subset \bigcup_{i \in I} Q'_i$ if and only if $P = \bigcup_{i \in I} Q_i$.*

Note that these modifications are only used in the analysis; we don't need to produce them algorithmically. We now prove Lemma 2.

Proof of Lemma 2. Let $\mathcal{S} = \mathcal{Q} \cup \mathcal{R}$ be the union of the global optimum and the local optimum. Using Lemma 3, we obtain a new set $\mathcal{S}' = \mathcal{Q}' \cup \mathcal{R}'$ of properly non-piercing pieces and a finite set of points X such that a subset of $\mathcal{S}'$ covers X if and only if the corresponding subset of $\mathcal{S}$ covers P. Since $\mathcal{Q}$ and $\mathcal{R}$ both cover P, $\mathcal{Q}'$ and $\mathcal{R}'$ must both cover X.

Recall that $q = |\mathcal{Q}| = |\mathcal{Q}'|$ and $r = |\mathcal{R}| = |\mathcal{R}'|$. If $r \geq (1+\varepsilon)q$ for some $1 \geq \varepsilon > 0$, then by Lemma 1, we can replace a set of at most $C\frac{1}{\varepsilon^2}$ of the pieces in $\mathcal{R}'$ with a strictly smaller set of pieces from $\mathcal{Q}'$ to obtain another cover of X. If we swap the corresponding pieces in $\mathcal{R}$ with the corresponding pieces in $\mathcal{Q}$, then we get a small cover of P, proving the claim. ◀

Lemma 2 gives sufficient conditions for a small cover to be a $(1+\varepsilon)$-approximation of the optimum. We say that a small cover R is a *k-local optimum* if no set of at most k pieces from $\mathcal{R}$ can be replaced by a strictly smaller set of small pieces.

▶ **Corollary 4.** *There is some constant C' such that if P is a polygon and $\mathcal{R}$ is a small cover that is a k-local optimum for $k \geq C'$ and has r complete pieces, then r is at most $1 + \frac{C'}{\sqrt{k}}$ times the size of an optimal cover of P.*

Proof. Taking C to be the constant from Lemma 2, let $C' = \sqrt{C}$ and $k_0 = \lceil C' \rceil$. Because $\mathcal{R}$ is k-local optimum, no set of at most k pieces can be replaced by a smaller set of small pieces. If $r \leq k$, then $\mathcal{R}$ is necessarily an optimal solution and the claim holds. So suppose that $r > k$. Let $\mathcal{Q}$ be an optimal small cover of P with q complete pieces. Suppose for contradiction that $r \geq \left(1 + \frac{C'}{\sqrt{k}}\right) q$. Note that for $\varepsilon := \frac{C'}{\sqrt{k}} \leq \frac{C'}{\sqrt{k_0}} \leq 1$, Lemma 2 guarantees a set of $k' \leq C/\varepsilon^2 = C\frac{k}{C'^2} = k$ pieces can be replaced by fewer pieces from $\mathcal{Q}$. A contradiction to the fact that $\mathcal{R}$ is a k-local optimum. Hence, $r < \left(1 + \frac{C'}{\sqrt{k}}\right) q$. ◀

2.2 Polygons with Small Diameter

In this section, we describe a PTAS for small covers of polygons that intersect at most D squares of the integer grid. In order to obtain a PTAS, it is sufficient by Corollary 4 to show that, for any fixed k, a k-local optimum can be found in polynomial time. The strategy is as follows: we start with some small cover of P, and then test every set of k pieces in that cover to see if it can be replaced by a smaller set. If such a swap is possible, then we perform the swap and start over with the smaller cover. This repeats until no swap can be found.

The polygon P with n vertices intersects at most D integer grid squares, so we can find an initial cover with at most Dn squares–note that the intersection of a unit square with P has at most n connected components as each side of P is incident to at most one connected component. So finding a k-local optimum requires at most a polynomial (in n and D) number of swaps, because the number of pieces decreases in each step. It remains to show how to find a swap, or determine that none exists.

Given a set of complete small pieces (*not* necessarily a cover) in P, we define a corresponding "combinatorial structure". The combinatorial structure is determined by:

- The line arrangement containing edges of P and edges of the squares bounding the pieces.
- For each bounding square S, the connected component of $P^\circ \cap S$ defining the piece (indicated with a number).

Let $\mathcal{L}$ be the set of lines containing edges of P or edges of a bounding square. For each $(\ell_1, \ell_2, \ell_3) \in \mathcal{L}^3$, the first piece of information says on which side of ℓ_3 the intersection of ℓ_1 and ℓ_2 occurs, or if the intersection is contained in ℓ_3. (This is the orientation of the triangle formed by (ℓ_1, ℓ_2, ℓ_3) or the order type of the dual of the lines.) The slope of each line in $\mathcal{L}$ is

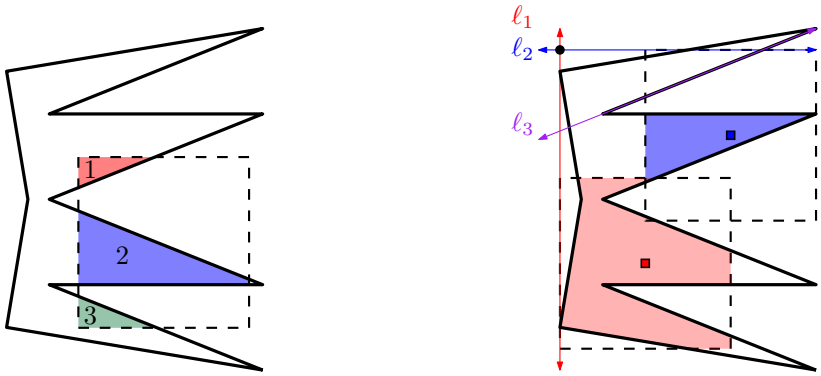


Figure 3 The combinatorial structure of a set of complete small pieces. Left: The connected components of $P^\circ \cap S$ are numbered. Right: The centers of the bounding squares of the two pieces are marked with squares. The intersection of lines ℓ_1 and ℓ_2 lies above ℓ_3..

fixed, so all the information about the arrangement of $\mathcal{L}$ can be described by equalities and (strict) inequalities that are linear in the coordinates of the bounding squares. For example, if ℓ_1 is a vertical line containing the left edge of a square with center (x_i, y_i), ℓ_2 is a horizontal line containing the top edge of a square with center (x_j, y_j), and $\ell_3 = \{(x, y) : \alpha x + \beta y = \gamma\}$ is a line containing an edge of P, then the combinatorial data contains one of:

- $\alpha(x_i - \frac{1}{2}) + \beta(y_j + \frac{1}{2}) = \gamma$,
- $\alpha(x_i - \frac{1}{2}) + \beta(y_j + \frac{1}{2}) < \gamma$, or
- $\alpha(x_i - \frac{1}{2}) + \beta(y_j + \frac{1}{2}) > \gamma$,

specifying on which side of ℓ_3 the intersection $(x_i - \frac{1}{2}, y_j + \frac{1}{2})$ of ℓ_1 and ℓ_2 occurs; see Figure 3.

A realization of a combinatorial structure is a set of pieces satisfying the constraints. Since the constraints are linear, the set of realizations of any fixed combinatorial structure is a convex set. Note that the combinatorial data does not ensure that a realization corresponds to a cover of the polygon P. For instance, it is easy to make an example where all the squares are placed almost on top of each other, while big regions of P remain uncovered. Given just the combinatorial data of a (hypothetical) arrangement, we can use linear programming (LP) to check if it can actually be realized by some set of small pieces. What we really want to be able to check is if *any* such realization corresponds to a cover of P. The following lemma says that we only need to check a single realization.

▶ **Lemma 5.** *For a given combinatorial structure, either every or no realization is a cover.*

Proof. Let A and B be realizations of a combinatorial structure with m squares. We show that A is a cover if and only if B is a cover. Suppose that A is a cover. We continuously move the bounding squares from their positions in A to their positions in B, along straight lines in the $2m$-dimensional coordinate space. If, during this process, the arrangement stops being a cover, then one of the following cases occurs: (a) part of a polygon edge is uncovered, (b) part of the interior is uncovered, or (c) a piece becomes disconnected. The cases are illustrated in Figure 4. We consider a snapshot right after the first point becomes uncovered. In case (a), the uncovered region is a triangle (or a rectangle). Note that the intersection of a right angle of this region has changed sides with respect to the polygon side independent of whether both or just one involved square is moving. In case (b), the uncovered region is a rectangle and its appearance is due to two squares that become disjoint due to the movement of at least one square. In case (c), a side of a moving square now intersects a side of the polygon which was previously disjoint. Consequently, in each case some of the combinatorial

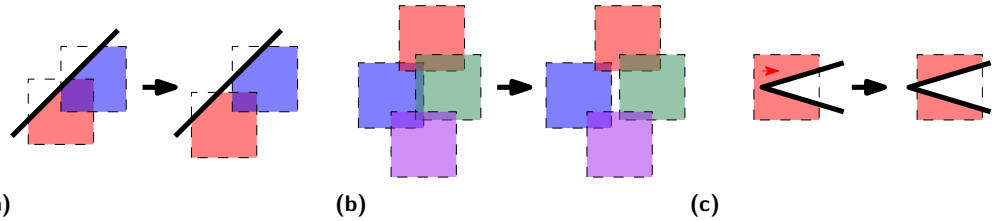


Figure 4 If the bounding squares are moving continuously, then there are three ways that a covering might stop being valid. Any of these change the arrangement of lines in $\mathcal{L}$. (a) part of an edge stops being covered. (b) part of the interior stops being covered. (c) a piece ceases to be connected.

data is changing. Since the set of realizations of a combinatorial structure is convex, we conclude that this doesn't happen. Since the arrangement forms a cover at every point in the motion, we conclude that B must be a cover.

By a symmetric argument, it is clear that if B is a cover, then A is a cover. ◀

Given a cover with complete pieces, we want to check if some set of k of its pieces can be swapped for a set of $k-1$ pieces. This is accomplished by the next lemma.

▶ **Lemma 6.** *Let P be a polygon with n vertices and consider a small cover $\mathcal{R}$ of P with m pieces. For each fixed k, we can determine in $\mathcal{O}_k((n+m)^{O(k)})$ time whether any set of k of the pieces from the cover can be replaced by $k-1$ small pieces (not necessarily from the cover) to form a cover with fewer pieces.*

Proof. For each of the $\binom{m}{k}$ k-subsets of $\mathcal{R}$, we remove it and test each of the ways of adding $k-1$ squares to the combinatorial arrangement. We need to argue that for any fixed k, there are only polynomially many ways of adding $k-1$ squares to a combinatorial structure with m squares. This corresponds to adding $2k$ vertical lines and $2k$ horizontal lines to the arrangement. A line arrangement with ℓ lines contains at most $O(\ell^2)$ intersection points. Thus there are $O(\ell^2)$ combinatorially different ways to add a new line with a fixed slope. A combinatorial structure with m squares has $n+2m$ lines. So the number of ways of adding the new $4k$ lines is $\mathcal{O}_k((n+m)^{4k})$. The intersection of a square with P can have at most n connected components, so in total there are $\mathcal{O}_k((n+m)^{4k+1})$ ways of adding k squares to the arrangement. For fixed k, this is polynomial in n and m.

For each of these candidate combinatorial structures, we need to check whether it represents a valid covering. By Lemma 5, it suffices to find a single representative set of pieces (if one exists). Each element of the combinatorial data is a linear constraint involving the coordinates of the squares and either an equality or a strict inequality. We construct an LP instance from these linear constraints, adding a new variable t and replacing each strict inequality $\mathcal{C} > 0$ with $\mathcal{C} \geq t$. To test if a solution exists, we solve the LP instance in which the objective function is to maximize t. If there is no feasible solution with $t > 0$, then we conclude that there is no valid swap with this combinatorial structure. If we find a feasible solution with $t > 0$, then we check if this solution covers all of P. We do this by constructing the pieces explicitly and computing their union. If the union does not cover P, then we know that no arrangement with this combinatorial structure does. The only free variables in the LP instance are t and the $2(k-1)$ coordinates of the newly-added bounding squares. So in order to solve the LP instance, we can just check all of the $(n+m)^{\mathcal{O}(k)}$ basic solutions. This takes (strongly-)polynomial time for a fixed k.

We should check that this procedure finds a swap if one exists. Suppose that there is a set of k pieces in the initial cover that can be swapped with $k-1$ pieces in order to obtain a new cover $\mathcal{R}'$. The cover $\mathcal{R}'$ has a combinatorial structure, which is one of the candidate structures tested by our procedure. Furthermore, $\mathcal{R}'$ is a realization of this structure with the positions of the $m-k$ old pieces being the same as in $\mathcal{R}$, so such a realization exists, and the linear programming step will find one (but the realization found is not necessarily the same one as $\mathcal{R}'$). Since $\mathcal{R}'$ is a cover, by Lemma 5, the realization found will be a cover. ◀

▶ **Theorem 7.** *There is an algorithm that, given a polygon P and a parameter ε, computes a $(1+\varepsilon)$-approximation of the optimal small cover of P. The algorithm runs in time $(Dn)^{poly(\frac{1}{\varepsilon})}$, where D is the number of integer grid squares intersecting P.*

Proof. Fix k such that $\frac{C'}{\sqrt{k}} \leq \varepsilon$, and $k \geq C'$, where C' is the constant from Corollary 4.

We can construct a cover of P with small pieces by overlaying P with the integer grid and taking the connected components of each intersection. Recall that the intersection of a unit square with P has at most n connected components. Hence, we obtain an initial solution with at most Dn pieces, where D is the number of integer grid squares intersecting P. We then check if this cover is a k-local optimum using the procedure from Lemma 6.

If this cover is not a k-local optimum, then we perform the swap and repeat the process until no more valid swaps remain. Every swap decreases the total number of pieces by 1, so the algorithm terminates after at most Dn steps, and when it does we have obtained a cover such that no set of k pieces can be swapped for a strictly smaller set of pieces. By Corollary 4, this solution has at most $(1+\varepsilon)$-times as many pieces as an optimal solution. It is clear by Lemma 6 that the entire process takes time at most $(Dn)^{\mathrm{poly}(\frac{1}{\varepsilon})}$. ◀

The algorithm in Theorem 7 runs in polynomial time on real RAM, but the bit-complexity of the coefficients can grow by a constant factor after each swap, potentially leading to exponential coefficient growth. In order to obtain an algorithm that runs in strongly-polynomial time, we can adjust the coordinates after each swap. Let m be the number of squares of the cover. We consider the LP instance L corresponding to the combinatorial structure of the cover. Here, L has two variables for all m squares, not just the new squares introduced in the latest swap. The cover that we found is a feasible solution $x \in \mathbb{R}^{2m+1}$ to L. There are two coordinates in x for each square and one for the value t in the strict inequalities. We want to find a *basic* feasible solution to L, i.e., a vertex of the polyhedron defined by the constraints, because the bit-complexity of the variables in such a solution is polynomial in n, D and the bit-complexity of the coordinates of vertices of P. The value of t (the variable for strict inequalities) should still be positive in the new solution.

This can be done in strongly-polynomial time. We can write L such that all constraints have form $\mathcal{C} \geq 0$. We start by evaluating each constraint at the feasible solution x. A constraint $\mathcal{C} \geq 0$ is *tight* if $\mathcal{C}(x) = 0$. Let $\mathcal{D}$ be the set of tight constraints and let $\mathcal{H} = \{y \in \mathbb{R}^{2m+1} \mid \forall \mathcal{C} \in \mathcal{D} : \mathcal{C}(y) = 0\}$. In other words, $\mathcal{H}$ is the affine subspace that causes the tight constraints to be tight. The solution x is a basic feasible solution if $\mathcal{H}$ has dimension 0. Otherwise, we can find a direction $v \in \mathrm{Vec}(\mathcal{H})$ such that t does not decrease when moving in direction v. We move from x in direction v until an additional constraint becomes tight.

We obtain a new feasible solution x' to L, where t at x' is at least as large as t at x. Furthermore, the dimension of $\mathcal{H}'$, the subspace of tight constraints corresponding to x', has decreased by at least 1 compared to $\mathcal{H}$. So we can repeat this process to find a basic feasible solution with the required properties. The number of arithmetic operations depends only on the number of variables and the number of constraints, so this process runs in strongly polynomial time. The bit-complexity of a basic feasible solution is polynomial in n, D and the bit-complexity of the coordinates of vertices of P. In particular, it does not increase with the number of swaps performed by the algorithm in Theorem 7.

2.3 Polygons with Large Diameter

In this section, we assume the existence of an algorithm $\mathcal{A}$ providing a $(1+\varepsilon)$-approximation for a small cover of a polygon P with diameter at most $O(n/\varepsilon)$. We show that from such an algorithm, one can design an algorithm $\mathcal{A}'$ that for an arbitrary polygon P, for which the size of a minimum cover of P is OPT, can find a value λ such that $\text{OPT} \leq \lambda \leq (1+\varepsilon_0)\text{OPT}$, where $\varepsilon_0 \leq \alpha\varepsilon$ for some universal constant α. The algorithm also provides an implicit description of a cover with this approximation guarantee. Recall that we work in the real RAM model. In particular, we can perform basic arithmetic operations like addition, multiplication, and division on the coordinates of the corners of the polygon in constant time.

▶ **Theorem 8.** *Assume there exists an algorithm $\mathcal{A}$ which for any polygon P with n vertices and diameter $O(n/\varepsilon)$ finds a $(1+\varepsilon)$-approximate minimum cover of P in time $f(n,\varepsilon)$. Then there exists an algorithm $\mathcal{A}'$ that for any n-vertex polygon P returns a value λ_P with $OPT \leq \lambda_P \leq (1+\alpha\varepsilon)OPT$, where OPT is the size of a minimal cover of P with small pieces and α is a sufficiently large universal constant. The runtime of $\mathcal{A}'$ is $f(\alpha n,\varepsilon)\cdot poly(n/\varepsilon)$. The algorithm also produces an implicit description of a $(1+\alpha\varepsilon)$-approximate cover. For any query point $q \in P$, the algorithm can in time $O(poly(n/\varepsilon)+f(4,\varepsilon))$ return all pieces in the cover containing q.*

▶ **Corollary 9.** *There is an algorithm that, given a polygon P and a parameter $0<\varepsilon<1$, computes a $(1+\varepsilon)$-approximation to the number of pieces in an optimal small cover of P. The algorithm runs in time $(n/\varepsilon)^{poly(1/\varepsilon)}$.*

The algorithm also produces an implicit description of a $(1+\varepsilon)$-approximate cover. For any query point $q \in P$, the algorithm can in time $O(poly(n/\varepsilon)+(1/\varepsilon)^{poly(1/\varepsilon)}))$ return all pieces in the cover containing q.

Proof. Use Theorem 8 with $\mathcal{A}$ being the algorithm of Theorem 7 with $D = O((n/\varepsilon)^2)$. ◀

We now sketch the idea of the reduction. We first argue that there is a set of $k = O(n)$ disjoint "large" *corridors* $\mathcal{C} = \{C_1,\ldots,C_k\}$ with each $C_i \subset P$ such that we can solve the covering problem on $C_1,\ldots,C_k$, and $P \setminus \bigcup \mathcal{C}$ separately. Each corridor C_i is constructed to have a pair of sides that are either both vertical or both horizontal and are of distance $\geq 1/\varepsilon$ apart. The remaining two sides are both contained in an edge of P. Finally, each C_i is essentially picked to be of maximum volume satisfying these properties.

We will show that the union of $(1+O(\varepsilon))$-approximate covers of $C_1,\ldots,C_k$ and $P \setminus \bigcup \mathcal{C}$ yields a $(1+O(\varepsilon))$-approximate cover of P. For $P \setminus \bigcup \mathcal{C}$, the diameter of each connected component is at most n/ε, so we can use the algorithm $\mathcal{A}$ for bounded diameter polygons. The corridors are more tricky as the size of their optimal solutions cannot be bounded in terms of n, so for those, we maintain an implicit solution. Denote by a_i and p_i the area and perimeter of C_i. If $a_i \geq p_i/\varepsilon$, one can show that the intersection C_i with the integer grid gives an $1+O(\varepsilon)$ approximate cover, and this solution is easy to describe implicitly. The tricky part is handling a corridor C_i with $a_i \leq p_i/\varepsilon$ where C_i is "long" and "skinny". Suppose that C_i has a pair of vertical edges (the horizontal case is similar). In this case, we partition C_i further into corridor sections $(S_i^j)_{j=1}^t$, all having a pair of vertical edges and width $1/\varepsilon$. We then again argue that it suffices to solve the problem on each S_i^j with an approximation ratio of $1+O(\varepsilon)$. Unfortunately, we cannot afford to find a solution for each S_i^j separately, but we can argue about a certain monotonicity of sizes of optimal covers of the S_i^j. Namely, after appropriate translations, it is either the case that $S_i^j \subset S_i^{j+1}$ for all j or it is the case that $S_i^j \supset S_i^{j+1}$ for all j. In the former case, we show that the size of the optimal cover of S_i^j is at most the size of the optimal cover of S_i^{j+1} and a similar claim holds in the latter

case. Based on these properties, and defining $\ell = \varepsilon^4 t$, we show that it suffices to approximate the minimum covers of $S_i^{\ell}, S_i^{2\ell}, \dots S_i^{\lfloor t/\ell \rfloor \ell}$. For any S_i^j with $k\ell \le j < (k+1)\ell$, for some k, we can then use the size of the approximate cover of $S_i^{(k+1)\ell}$ as an upper bound on the size of the optimal cover of S_i^j. This only requires us to run $\mathcal{A}$ on the $t/\ell = O(1/\varepsilon^4)$ instances of the form $S_i^{(k+1)\ell}$. While for some values of k, this approach might not give $1 + O(\varepsilon)$ approximation to S_i^j, the monotonicity ensures that for most values of k, the approximation is good, and suffices to give a final approximation guarantee of $1 + O(\varepsilon)$.

2.4 Equivalence of Small Covers and Partitions

The proof of Lemma 3 lets us show the equivalence of small covering and partitioning – a result which may be interesting in its own right.

▶ **Theorem 10.** *For any polygon P, possibly with holes, optimal small covers and small partitions have the same number of pieces.*

Proof. Consider an optimal small cover $\mathcal{Q}$ consisting of complete small pieces; see Figure 5.

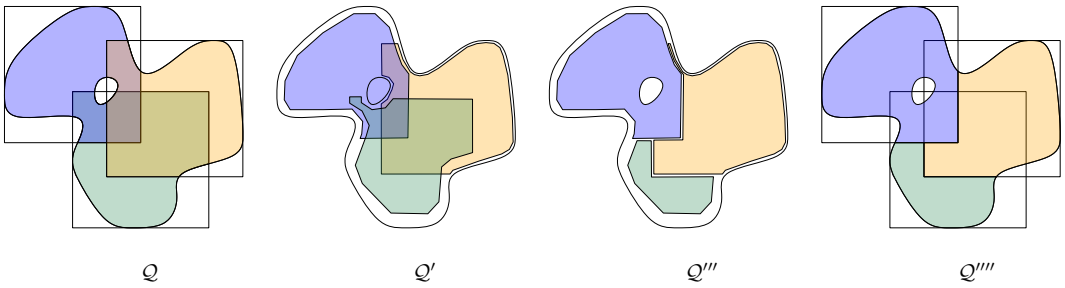


Figure 5 Steps in the proof of Theorem 10.

Consider the overlay of $\mathcal{Q}$ and choose a point in the interior of each cell which is also in P. Construct a set of proper non-piercing regions $\mathcal{Q}'$ as in Lemma 3. As described in the proof, the pieces in $\mathcal{Q}'$ are subsets of those in $\mathcal{Q}$, except that some holes of P have been filled. Importantly, the boundaries of the pieces in $\mathcal{Q}'$ are just slight perturbations of those in $\mathcal{Q}$.

The following process of *lens bypassing* is described in [34, Section 4]. Since $\mathcal{Q}'$ are non-piercing, any intersection $Q_i' \cap Q_j'$ consists of a set of *lenses*, which are regions bounded by one interval of the boundaries of each set Q_i' and Q_j'. In the process of lens bypassing, we consider an inclusion-wise minimal lens ℓ, say in the intersection $Q_i' \cap Q_j'$. Let the boundary of ℓ be $\gamma_i \cup \gamma_j$, where γ_i and γ_j are on the boundaries of Q_i' and Q_j', respectively. We change the boundary of Q_i' by replacing γ_i by a curve following γ_j (but slightly outside ℓ to avoid non-proper intersections). This effectively removes ℓ from Q_i'. It is shown [34, Lemma 4.6] that the operation results in a set of regions that are also non-piercing. Repeating the process results in a set $\mathcal{Q}''$ of pairwise disjoint regions that are subsets of the original regions $\mathcal{Q}'$ and still contain the points X.

Consider any hole H of P, i.e., H is a bounded connected component in the complement of P. Recall that in the pieces $\mathcal{Q}'$, the hole H might have been filled in some of them, so H might also be filled in some of the pieces $\mathcal{Q}''$. In our modifications from Q to Q'', we have only ever made modifications along curves that are disjoint from H. Hence, for each piece $Q_i'' \in \mathcal{Q}''$, we either have $H \subset Q_i''$ (H is filled) or $H \cap Q_i'' = \emptyset$ (H is still a hole of Q_i'' or outside the outer boundary). We can remove H from all pieces where it is filled, thus reintroducing H as a hole, and this will not break connectivity. We do so for all holes of P.

We end up with regions $\mathcal{Q}'''$ that are pairwise disjoint and all contained in P. Furthermore, the regions $\mathcal{Q}'''$ still cover P in the sense that each region in the overlay of the original cover $\mathcal{Q}$ is contained in a region in the overlay of $\mathcal{Q}'''$, except that the boundary might be slightly perturbed. We can thus create pieces $\mathcal{Q}''''$ analogous to $\mathcal{Q}'''$ using the original boundaries, and these pieces will cover all of P and still be pairwise interior-disjoint. We therefore obtain a small partition of P. ◀

3 Hardness for Small-Cover in 3 dimensions

In this section, we give a gap-preserving reduction from Set-Cover to 3-dimensional small cover. The best-known inapproximability result for Set-Cover is the following:

▶ **Theorem 11.** *(Dinur and Steurer [22]) For any $\alpha > 0$, it is NP-hard to approximate an instance of Set-Cover of size n to within a factor of $(1-\alpha)\ln(n)$.*

Our results show that Small-Cover in 3 dimensions is no easier to approximate than Set-Cover.

▶ **Theorem 12.** *Let I be an instance of Set-Cover where the universe is $U = \{1, \ldots, n\}$ and the sets are $\mathcal{S} = \{S_1, \ldots, S_m\}$. Then we can construct in polynomial time a simple polyhedron $P \subseteq \mathbb{R}^3$ with $\mathcal{O}\left(n + \sum_i |S_i|\right)$ vertices such that P has a unit cube cover of size $k+5$ if and only if U can be covered by k of the sets in $\mathcal{S}$. The vertices of P have bit-complexity at most $\mathcal{O}(\log(nm))$ and P is contained in a box of size $1 \times 2 \times 3$.*

Proof. For simplicity, we will scale the coordinates by a factor of 8. So the polyhedron P will have a cover with pieces that fit inside $8 \times 8 \times 8$-cubes. We will construct P from a union of flat plane pieces, but it can easily be thickened to obtain a non-degenerate polyhedron.

To construct P, we start by making layers of *hook* pieces. There is one type of hook piece for each of the m sets in $\mathcal{S}$. The hook piece for S_j is shown in Figure 6a. For each $i \in S_j$, we place a copy of the corresponding hook piece at z-coordinate $\frac{mi+j}{mn}$.

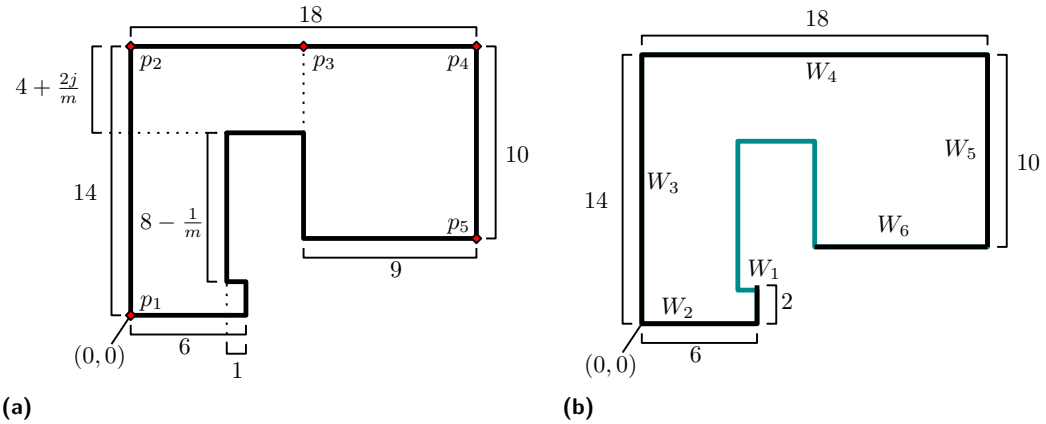


Figure 6 (a) Specification of a hook piece. The (x, y)-positions of the points p_1 through p_5 are also shown. (b) The segments marked show the wall pieces that connect the hook pieces together. Segments W_1 through W_5 extend to cover the height of the entire figure, while segment W_6 is extended over n disjoint intervals. The outline of a hook piece is shown in green.

The hook pieces are connected by a wall, which is formed by the segments marked W_1 through W_5 in Figure 6b extruded to cover the z-coordinates $[\frac{m+1}{mn}, \frac{(m+1)n}{mn}]$. Additionally, the segment marked W_6 is extruded to cover the z coordinates $[\frac{mi+1}{mn}, \frac{(m+1)i}{mn}]$ for each $i \in U$. The entire construction is illustrated in Figure 7.

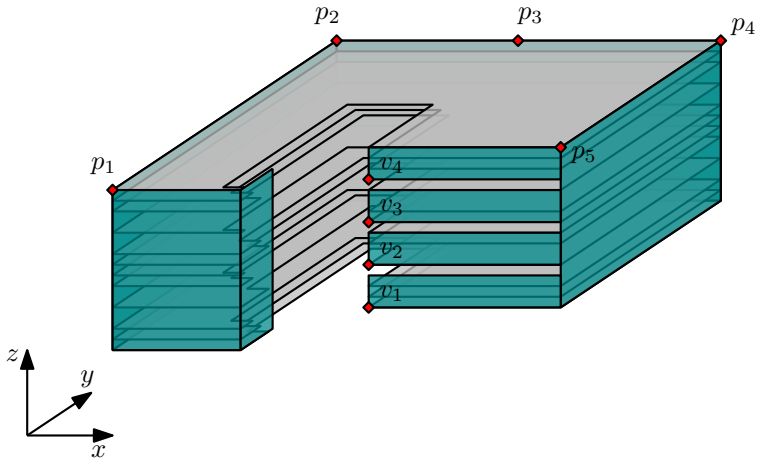


Figure 7 A rendering of the polyhedron P. The specific instance shown corresponds to $n = 4$, $m = 4$, $S_1 = \{1, 2, 3\}$, $S_2 = \{1, 3, 4\}$, $S_3 = \{1, 2, 4\}$, and $S_4 = \{2, 3, 4\}$. The hook pieces are shown in gray and the wall pieces are shown in green. The vertical scale is exaggerated for clarity.

Suppose that U can be covered by k of the sets in $\mathcal{S}$. We want to show that P can be covered with $k + 5$ pieces that fit inside an $8 \times 8 \times 8$-unit cube. First note that the 5 cubes shown in Figure 8 are sufficient to cover most of P.

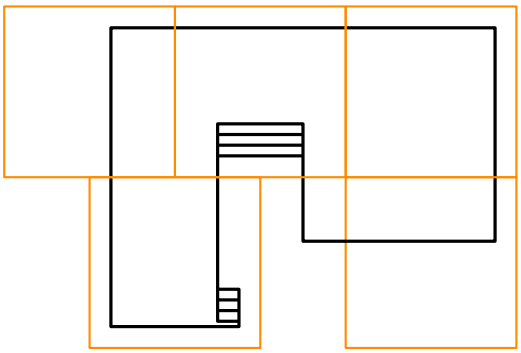


Figure 8 These 5 cubes cover most of P.

The intersection of P with each of these cubes is connected. There are n connected chunks remaining uncovered, corresponding to the n elements of U. Let Q_i be the connected part of the uncovered region containing v_i. For each S_j, there is a connected piece containing $\bigcup_{i \in S_j} Q_i$ that fits inside the cube $[4, 12] \times [2 - \frac{4j-1}{2m}, 10 - \frac{4j-1}{2m}] \times [0, 8]$, as illustrated in Figure 9a and 9b. So if U can be covered by k of the S_j, then P can be covered by $k + 5$ pieces.

Now suppose that P can be covered by $k + 5$ pieces. We first designate five points $p_1, \ldots, p_5$ (as shown in Figure 6a and 7). No $8 \times 8 \times 8$ cube can contain more than 1 of these points, so there are at least five pieces in the covering, one for each of these points. These pieces also cannot contain any of the points $v_1, \ldots, v_n$, so the remaining pieces must cover these points.

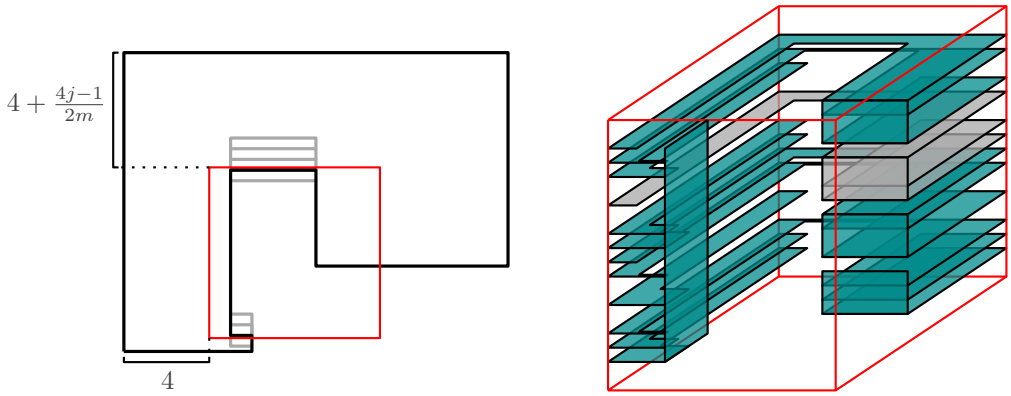


Figure 9 (a) The intersection of the cube $[4, 12] \times [2 - \frac{4j-1}{2m}, 10 - \frac{4j-1}{2m}] \times [0, 8]$ with a hook piece corresponding to S_j connects some of the Q_i. (b) A 3-dimensional view of the intersection of P with the cube $[4, 12] \times [2 - \frac{4j-1}{2m}, 10 - \frac{4j-1}{2m}] \times [0, 8]$. Shown is the example from Figure 7 with $j = 3$. The green part is connected and contains Q_1, Q_2 and Q_4. The gray part contains Q_3, but cannot be connected to the green part without leaving the cube.

Suppose that some piece Q in the cover contains $\{v_i : i \in S\}$ for some set $S \subseteq U$. We want to show that $S \subseteq S_j$ for some j. By assumption, each $i \in U$ is in some S_j, so WLOG suppose $|S| \geq 2$. Let $i, \ell \in S$ with $i \neq \ell$. So Q contains a path γ from v_i to v_ℓ. No part of the walls W_3 through W_5 is close enough to v_i to be contained in Q, so γ must connect through W_1 or W_2.

In order to get from v_i to W_1 or W_2, γ must go through some hook piece, say this hook piece corresponds to S_j. Such a connection contains y coordinates covering the interval $[2 - \frac{2j}{m}, 10 - \frac{2j-1}{m}]$ (see Figure 9a and 9b). Since Q is contained in an $8 \times 8 \times 8$-unit cube, it cannot contain points with y-coordinates covering $[2 - \frac{2j'}{m}, 10 - \frac{2j'-1}{m}]$ for any $j' \neq j$. So any v_ℓ contained in Q is connected to v_i through a hook piece corresponding to S_j. This is only possible if $\ell \in S_j$. So $S \subseteq S_j$.

Thus, U can be covered by k of the S_j if and only if P can be covered by $k + 5$ connected pieces each fitting inside an $8 \times 8 \times 8$-unit cube.

If we thicken P slightly, we obtain a simple (topologically ball-shaped) polyhedron. To see this, we observe that P is contractable. Indeed, each hook piece (strongly) deformation retracts onto walls W_1 through W_6. So all of the hook pieces deformation retract onto the wall pieces. Since the union of the wall pieces is contractable, P is contractable. ◀

We note that it is possible to trim the cover of size $k + 5$ described in the proof in order to obtain a partition, so we get hardness for both Small-Cover and Small-Partition. In fact, it is easy to see that the problems are equivalent in 3 and higher dimensions: If $Q_1, \ldots, Q_q$ form a cover for a polyhedron P, we can define $Q'_i = Q_i \setminus \bigcup_{j=1}^{i-1} Q_j$ for $i = 1, \ldots, q$. Then the pieces Q'_i are pairwise disjoint and also cover P, but they may be disconnected. However, if Q'_i is disconnected, we can connect the parts by thin tunnels, staying in the original piece Q_i so we don't violate the size constraint. These tunnels are removed from the other pieces Q'_j, and choosing them sufficiently thin, they will not disconnect any piece.

▶ **Corollary 13.** *In 3 dimensions, for any $\alpha > 0$, it is NP-hard to approximate Small-Cover and Small-Partition better than $(1 - \alpha) \ln(n)$, where n is the number of bits needed to describe the input polyhedron. This holds even when the polyhedron is simple and contained in a box of size $1 \times 2 \times 3$.*

References

1 Anders Aamand, Mikkel Abrahamsen, Reilly Browne, Mayank Goswami, Prahlad Narasimhan Kasthurirangan, Linda Kleist, Joseph S. B. Mitchell, and Jack Polishchuk, Valentin Stade. Covering and partitioning complex objects with small pieces, 2026. arXiv:2603.23216.

2 Mikkel Abrahamsen. Covering polygons is even harder. In *Symposium on Foundations of Computer Science (FOCS)*, pages 375–386. IEEE, 2021. doi:10.1109/FOCS52979.2021.00045.

3 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. *J. ACM*, 69(1):4:1–4:70, 2022. doi:10.1145/3486220.

4 Mikkel Abrahamsen, Joakim Blikstad, André Nusser, and Hanwen Zhang. Minimum star partitions of simple polygons in polynomial time. In *Symposium on Theory of Computing, (STOC)*, pages 904–910. ACM, 2024. doi:10.1145/3618260.3649756.

5 Mikkel Abrahamsen and Nichlas Langhoff Rasmussen. Partitioning a polygon into small pieces. In *Symposium on Discrete Algorithms (SODA)*, pages 3562–3589. SIAM, 2025. doi:10.1137/1.9781611978322.118.

6 Mikkel Abrahamsen and Jack Stade. Hardness of packing, covering and partitioning simple polygons with unit squares. *SIAM Journal on Computing*, pages FOCS24–29, 2026. doi:10.1137/24M171824X.

7 Esther M. Arkin, Rathish Das, Jie Gao, Mayank Goswami, Joseph S. B. Mitchell, Valentin Polishchuk, and Csaba D. Tóth. Cutting polygons into small pieces with chords: Laser-based localization. In *European Symposium on Algorithms (ESA)*, pages 7:1–7:23, 2020. doi:10.4230/LIPIcs.ESA.2020.7.

8 Rom Aschner, Matthew J. Katz, Gila Morgenstern, and Yelena Yuditsky. Approximation schemes for covering and packing. In *WALCOM: Algorithms and Computation*, pages 89–100, 2013. doi:10.1007/978-3-642-36065-7_10.

9 Aaron T. Becker, Sándor P. Fekete, Alexander Kröller, SeoungKyou Lee, James McLurkin, and Christiane Schmidt. Triangulating unknown environments using robot swarms. In *Symposium on Computational Geometry (SoCG)*, pages 345–346, 2013. doi:10.1145/2462356.2462360.

10 A. Bezdek and K. Bezdek. A solution of conway's fried potato problem. *Bulletin of the London Mathematical Society*, 27(5):492–496, 1995. doi:10.1112/blms/27.5.492.

11 A. Bezdek and K. Bezdek. Conway's fried potato problem revisited. *Arch. Math.*, 66(6):522–528, 1996. doi:10.1007/BF01268872.

12 Karol Borsuk. Drei Sätze über die n-dimensionale euklidische Sphäre. *Fundamenta Mathematicae*, 20(1):177–190, 1933. doi:10.4064/fm-20-1-177-190.

13 Maike Buchin and Leonie Selbach. Decomposing polygons into fat components. In *Canadian Conference on Computational Geometry (CCCG)*, pages 175–184, 2021.

14 Antonio Cañete, Isabel Fernández, and Alberto Márquez. Conway's fried potato problem: a (quadratic) algorithm leading to an optimal division for convex polygons. In *Discrete Mathematics Days (DMD)*, pages 71–76, 2022.

15 Bernard Chazelle. Approximation and decomposition of shapes. In *Algorithmic and Geometric Aspects of Robotics*, volume 1 of *Advances in Robotics*, pages 145–185. L. Erlbaum Associates Inc., 1987.

16 Bernard Chazelle and David P. Dobkin. Optimal convex decompositions. In *Computational Geometry*, volume 2 of *Machine Intelligence and Pattern Recognition*, pages 63–133. Elsevier, 1985. doi:10.1016/B978-0-444-87806-9.50009-8.

17 Bernard Chazelle and Leonidas Palios. Decomposition algorithms in geometry. In *Algebraic Geometry and its applications*, pages 419–447. Springer, 1994. doi:10.1007/978-1-4612-2628-4_27.

18 Xun Chen and Mingyong Chen. Skeleton partition models for 3D printing. In *Second International Conference on Medical Imaging and Additive Manufacturing (ICMIAM 2022)*, 2022. doi:10.1117/12.2636369.

19 Hallard T. Croft, Kenneth J. Falconer, and Richard K. Guy. *Unsolved Problems in Geometry*. Problem Books in Mathematics: Unsolved Problems in Intuitive Mathematics. Springer, 1991. doi:10.1007/978-1-4612-0963-8.

20 Mirela Damian and Sriram V. Pemmaraju. Computing optimal diameter-bounded polygon partitions. *Algorithmica*, 40(1):1–14, 2004. doi:10.1007/s00453-004-1092-3.

21 Mark de Berg, Sándor Kisfaludi-Bak, Morteza Monemizadeh, and Leonidas Theocharous. Clique-based separators for geometric intersection graphs. *Algorithmica*, 85(6):1652–1678, 2023. doi:10.1007/S00453-022-01041-8.

22 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing (STOC)*, pages 624–633. ACM, 2014. doi:10.1145/2591796.2591884.

23 Sándor P. Fekete, Tom Kamphans, Alexander Kröller, Joseph S. B. Mitchell, and Christiane Schmidt. Exploring and triangulating a region by a swarm of robots. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX-RANDOM)*, pages 206–217. Springer, 2011. doi:10.1007/978-3-642-22935-0_18.

24 Sariel Har-Peled and Kent Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. *SIAM Journal on Computing*, 46(6):1712–1744, 2017. doi:10.1137/16M1079336.

25 Thomas Jenrich and Andries E. Brouwer. A 64-dimensional counterexample to Borsuk's conjecture. *Electron. J. Comb.*, 21(4):4, 2014. doi:10.37236/4069.

26 Xiaotong Jiang, Xiaosheng Cheng, Qingjin Peng, Luming Liang, Ning Dai, Mingqiang Wei, and Cheng Cheng. Models partition for 3D printing objects using skeleton. *Rapid Prototyping Journal*, 23:54–64, 2017. doi:10.1108/RPJ-07-2015-0091.

27 Jeff Kahn and Gil Kalai. A counterexample to Borsuk's conjecture. *Bulletin of the American Mathematical Society*, 29(1):60–62, 1993. doi:10.1090/S0273-0979-1993-00398-7.

28 J. Mark Keil. Polygon decomposition. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of computational geometry*, chapter 11, pages 491–518. Elsevier, 1999. doi:10.1016/B978-044482537-7/50012-7.

29 J. Mark Keil and Jörg-R. Sack. Minimum decompositions of polygonal objects. In *Computational Geometry*, volume 2 of *Machine Intelligence and Pattern Recognition*, pages 197–216. Elsevier, 1985. doi:10.1016/B978-0-444-87806-9.50012-8.

30 Minseok Ko, Yeongjun Yoon, Jaeyeon Kim, Samyeon Kim, and Soonjo Kwon. D-ECOmposer: Sustainable part decomposition for additive manufacturing using machine learning based life cycle assessment. *Additive Manufacturing*, 103:104759, 2025. doi:10.1016/j.addma.2025.104759.

31 Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. Chopper: Partitioning models into 3D-printable parts. *ACM Trans. Graph.*, 31(6):129:1–129:9, 2012. doi:10.1145/2366145.2366148.

32 Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

33 Joseph O'Rourke, Subash Suri, and Csaba D. Tóth. Polygons. In J. E. Goodman and J. O'Rourke, editors, *Handbook of discrete and computational geometry*, chapter 30, pages 787–810. Taylor & Francis, third edition edition, 2018. doi:10.1201/9781315119601.

34 Aniket Basu Roy, Sathish Govindarajan, Rajiv Raman, and Saurabh Ray. Packing and covering with non-piercing regions. *Discret. Comput. Geom.*, 60(2):471–492, 2018. doi:10.1007/S00454-018-9983-2.

35 Thomas C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992. doi:10.1109/5.163407.

36 Juraj Vanek, Jorge A. Garcia Galicia, Bedrich Benes, Radomír Mech, Nathan A. Carr, Ondrej Stava, and Gavin S. P. Miller. Packmerger: A 3D print volume optimizer. *Comput. Graph. Forum*, 33(6):322–332, 2014. doi:10.1111/CGF.12353.

37 Chris Worman. Decomposing polygons into diameter bounded components. In *Canadian Conference on Computational Geometry (CCCG)*, pages 103–106, 2003. URL: http://www.cccg.ca/proceedings/2003/9.pdf.

38 Yuan Yao, Longyu Cheng, and Zhengyu Li. A comparative review of multi-axis 3D printing. *Journal of Manufacturing Processes*, 120:1002–1022, 2024. doi:10.1016/j.jmapro.2024.04.084.

On the Maximum Number of Tangencies Among 1-Intersecting Curves

Eyal Ackerman
Department of Mathematics, Physics and Computer Science, University of Haifa at Oranim, Israel

Balázs Keszegh
HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary
ELTE Eötvös Loránd University, Budapest, Hungary

Abstract

According to a conjecture of Pach, there are $O(n)$ tangent pairs among any family of n Jordan arcs in which every pair of arcs has precisely one common point and no three arcs share a common point. This conjecture was proved for two special cases, however, for the general case the currently best upper bound is only $O(n^{7/4})$. This is also the best known bound on the number of tangencies in the relaxed case where every pair of arcs has *at most* one common point. We improve the bounds for the latter and former cases to $O(n^{5/3})$ and $O(n^{3/2})$, respectively. We also consider a few other variants of these questions, for example, we show that if the arcs are *x-monotone*, each pair intersects at most once and their left endpoints lie on a common vertical line, then the maximum number of tangencies is $\Theta(n^{4/3})$. Without this last condition the number of tangencies is $O(n^{4/3}(\log n)^{1/3})$, improving a previous bound of Pach and Sharir. Along the way we prove a graph-theoretic theorem which extends a result of Erdős and Simonovits and may be of independent interest.

2012 ACM Subject Classification Mathematics of computing → Extremal graph theory; Mathematics of computing → Graphs and surfaces

Keywords and phrases tangency graph, forbidden subgraph, extremal graph

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.2

Related Version *Full version*: `https://arxiv.org/abs/2603.11885` [2]

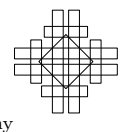
Funding *Balázs Keszegh*: Supported by the ERC Advanced Grant "ERMiD", no. 101054936 and by the EXCELLENCE-24 project no. 151504 Combinatorics and Geometry of the NRDI Fund.

Acknowledgements We are grateful to Rom Pinchasi for many helpful discussions on the problems studied in this paper. In particular he has suggested most of the ingredients of the proofs of Theorems 7 and 5 and later found out that similar results were already proved by Pach and Sharir [23]. We thank Balázs Patkós for his comments about Theorem 9. For the proof of Theorem 10 we used some back and forth interaction with Google's Large Language Model "Gemini" (`https://gemini.google.com/`).

1 Introduction

Arrangements of curves play an important role in Computational and Combinatorial Geometry with applications in domains such as robotics, computer graphics, computer vision, and combinatorial optimization [24]. Many interesting problems in these areas can be phrased as determining the maximum number of *tangencies* among curves under various restrictions. For example, occasionally, analyzing the complexity of an arrangement of curves boils down to estimating the number of faces of size two [5]. Such faces can often be perturbed into tangency points. Another example is Erdős's famous *Unit Distance Problem* which asks for the maximum number of pairs of points that are at unit distance from each other among n distinct points in the plane. This is equivalent to asking for the maximum number of tangencies among n unit circles in the plane.


© Eyal Ackerman and Balázs Keszegh;
licensed under Creative Commons License CC-BY 4.0




Throughout this paper we consider planar curves, that is, Jordan arcs, and assume that every two curves intersect in a finite number of points. A family of curves is *k-intersecting* if every two curves in it intersect in at most k points. Two curves are said to be *tangent* or *touching* if they have precisely one common point and this point is not a crossing point.[1] We will mainly consider the maximum number of tangencies among families of 1-intersecting curves. Such families are sometimes called *pseudo-segments*.[2] A family of curves is *precisely* 1-intersecting if every pair of curves has precisely one common point (either a crossing or a tangency point). We will always assume that no three curves share a common point, for otherwise, it is possible to have one point at which every pair of curves is tangent. Therefore, the number of tangent pairs is equal to the number of tangency points.

János Pach proposed the following nice conjecture.

▶ **Conjecture 1** (Pach [20]). *Every family of n precisely 1-intersecting planar curves in which no three curves share a common point admits $O(n)$ tangencies.*

Györgyi et al. [14] proved this conjecture for the very special case where the endpoints of the curves lie within a constant number of cells of their arrangement. Conjecture 1 was also settled by the authors for *x-monotone* curves [4]. Recall that a curve is *x-monotone* (resp., *bi-infinite* x-monotone) if it is the graph of some continuous function over some closed interval (resp., over $\mathbb{R}$). For arbitrary families of (precisely) 1-intersecting curves a non-trivial $O(n^{7/4})$ bound on the number of tangencies follows from a result of Keszegh and Pálvölgyi [17]:

▶ **Theorem 2** ([17]). *Every family of n k-intersecting planar curves admits $O_k\left(n^{2-\frac{1}{k+3}}\right)$ tangencies.*

We improve this bound for (precisely) 1-intersecting families as follows.

▶ **Theorem 3.** *Every family of n 1-intersecting planar curves admits $O(n^{5/3})$ tangencies.*

▶ **Theorem 4.** *Every family of n precisely 1-intersecting planar curves admits $O(n^{3/2})$ tangencies.*

Note that the maximum number of tangencies among n (x-monotone) 1-intersecting curves is $\Omega(n^{4/3})$. Indeed, it is not hard to construct a family admitting that many tangencies based on the famous construction of Erdős (see [21]) of n lines, n points and $\Theta(n^{4/3})$ point-line incidences (see the proof of Theorem 6).

For x-monotone 1-intersecting curves we have the following bound.

▶ **Theorem 5.** *Every family of n 1-intersecting x-monotone planar curves admits $O\left(n^{4/3}(\log n)^{1/3}\right)$ tangencies.*

This bound almost matches the above-mentioned $\Omega(n^{4/3})$ lower bound and improves by an $O\left((\log n)^{1/3}\right)$ factor a previous bound that follows from a result of Pach and Sharir [23]. For so-called *grounded* x-monotone 1-intersecting curves we have an asymptotically tight bound.

▶ **Theorem 6.** *The maximum number of tangencies among n x-monotone 1-intersecting planar curves whose left endpoints lie on a common vertical line is $\Theta(n^{4/3})$.*

---

[1] Here we follow the definition in, e.g., [22]. In other papers, e.g., [9], two curves may cross and also touch each other (at different points). However, for 1-intersecting curves the two definitions coincide.

[2] However, the term "pseudo-segments" is somewhat ill-suited in the context of tangencies, since straight-line segments can only be tangent at an endpoint.

The proofs of Theorems 5 and 6 rely on the following bounds for bi-infinite curves.

▶ **Theorem 7.** *The maximum number of tangencies among n bi-infinite x-monotone 1-intersecting curves is $\Theta(n \log n)$. In case of precisely 1-intersecting curves the maximum number of tangencies is $n-1$.*

The bound for bi-infinite x-monotone 1-intersecting curves was already proved by Pach and Sharir [23] using different and less elementary arguments. The following table summarizes all the above-mentioned results.

Table 1 Bounds on the maximum number of tangencies among n 1-intersecting curves.

	x-monotone	arbitrary
bi-infinite 1-intersecting	$\Theta(n \log n)$ [23] (Thm 7)	–
bi-inf. precisely 1-inters.	$n-1$ (Thm 7)	–
grounded 1-intersecting	$\Theta(n^{4/3})$ (Thm 6)	$\Omega(n^{4/3})$, $O(n^{3/2})$ (Cor 18)
1-intersecting	$\Omega(n^{4/3})$ [23], $O(n^{4/3}(\log n)^{1/3})$ (Thm 5)	$\Omega(n^{4/3})$, $O(n^{5/3})$ (Thm 3)
precisely 1-intersecting	$\Theta(n)$ [4]	$\Omega(n)$, $O(n^{3/2})$ (Thm 4)

1.1 A graph-theoretic theorem

For a graph H let $\mathrm{ex}(n, H)$ denote the maximum number of edges in an n-vertex *H-free* graph, that is, a graph which does not contain H as a subgraph. For a bipartite graph $H = (A \cup B, E)$ we denote by H^+ the graph we get by adding to H two new adjacent vertices a' and b' and connecting a' to every vertex in B and b' to every vertex in A. That is, $H^+ = ((A \cup \{a'\}) \cup (B \cup \{b'\}), E \cup \{(a', b')\} \cup \{(a', b) \mid b \in B\} \cup \{(a, b') \mid a \in A\})$. Erdős and Simonovits [10] proved the following result[3] when studying $\mathrm{ex}(n, Q_3)$, where Q_3 is the graph with 8 vertices and 12 edges that corresponds to a three-dimensional cube.

▶ **Theorem 8** ([10, Theorem 2])**.** *Let H be a bipartite graph with at least one edge such that $ex(n, H) = O(n^{2-\alpha})$ for some constant $0 \leq \alpha \leq 1$. Then $ex(n, H^+) = O\left(n^{2-\frac{\alpha}{\alpha+1}}\right)$.*

Observe that if $G = (A \cup B, E)$ is a bipartite H^+-free graph, then for every adjacent vertices $a \in A$ and $b \in B$ it holds that $G[(N_G(a) \setminus \{b\}) \cup (N_G(b) \setminus \{a\})]$ – the subgraph of G induced by their *open bineighborhood* – is H-free. Roughly speaking, if for every adjacent vertices in G the subgraph induced by their open bineighborhoods is not too dense because it is H-free, then it follows that G is also not too dense. We extend and refine Theorem 8 by showing that it holds whenever we have sparse subgraphs induced by open sub-bineighborhoods of pairs of (adjacent) vertices, be it for H-freeness or some other reason. To state the result we need the following definition: for a graph G and a nonnegative and nondecreasing function $f : \mathbb{R} \to \mathbb{R}$ we say that two vertices u and v have *f-sparse sub-bineighborhoods* if for every two disjoint subsets $U \subseteq N_G(u) \setminus \{v\}$ and $V \subseteq N_G(v) \setminus \{u\}$ it holds that $|E(G(U, V))| \leq f(|U \cup V|)$, where $G(U, V)$ is the bipartite subgraph $(U \cup V, \{(u, v) \in E(G) \mid u \in U \text{ and } v \in V\})$. For the proof of Theorem 3 we use the following theorem which extends Theorem 8 and may be of independent interest.

▶ **Theorem 9.** *Let G be a graph and let $f(n) = O(n^{2-\alpha})$ for some constant $0 \leq \alpha < 2$. If every pair of vertices in G has f-sparse sub-bineighborhoods, then $|E(G)| = O\left(|V(G)|^{2-\frac{\alpha}{\alpha+1}}\right)$. Moreover, if $\alpha \leq 1$, then it is enough that every pair of adjacent vertices has f-sparse sub-bineighborhoods for this bound to hold.*

[3] We present a simplified version of their result from which the original version can be easily deduced.

Clearly, Theorem 9 implies Theorem 8. Note that from Theorem 8 one can conclude by induction on t the classical Kővári-Sós-Turán Theorem [18] stating that an n-vertex $K_{t,t}$-free graph has $O_t\left(n^{2-\frac{1}{t}}\right)$ edges. It is a major open question in extremal combinatorics whether this bound is tight with positive answers only known for $t = 2$ and $t = 3$. Thus, the upper bound in Theorem 8 is tight for $\alpha = 1$ and $\alpha = \frac{1}{2}$, and, assuming that the bound of the Kővári-Sós-Turán Theorem is tight, for every $\alpha = \frac{1}{t}$ where t is a positive integer. Remarkably, for Theorem 9 we do have a matching lower bound for every $0 \leq \alpha \leq 1$.

▶ **Theorem 10.** *For every $0 \leq \alpha \leq 1$ there is a bipartite graph $G = (V, E)$ with $|E| = \Omega\left(|V|^{2-\frac{\alpha}{\alpha+1}}\right)$ edges such that every pair of vertices in G has f-sparse sub-bineighborhoods where $f(n) = O(n^{2-\alpha})$.*

1.2 Related work

Agarwal, Nevo, Pach, Pinchasi, Sharir and Smorodinsky [5] proved that any family of n pairwise intersecting x-monotone bi-infinite 2-intersecting curves admits at most $2n - 4$ tangencies. Families of 2-intersecting closed curves are called *pseudo-circles*. Erdős and Grünbaum [11] (see also [8, §7.1, Problem 14]) asked for the maximum number of tangencies among n pseudo-circles. An $\Omega(n^{4/3})$ lower bound for this problem follows from the many point-line incidences construction. The best upper bound is $O(n^{3/2})$ by a recent result of Janzer, Janzer, Methuku and Tardos [16]. As for pairwise intersecting pseudo-circles, a recent work with Damásdi, Pinchasi and Raffay [1] settled an old conjecture of Grünbaum [13] by showing that the maximum number of tangencies is $2n - 2$ in this case. A worse linear upper bound was proved in [5].

Ellenberg, Solymosi and Zahl [9] proved that any family of n plane algebraic curves of degree at most d admits $O_d(n^{3/2})$ tangencies.[4] If two curves may intersect an arbitrary number of times, then it is possible to have $\Omega(n^2)$ tangencies among n curves. Solving a conjecture of Richter and Thomassen it was shown by Pach, Rubin and Tardos [22] that if every pair of n closed curves intersects, then there are many more crossing points than tangency points. Specifically, they showed that if X denotes the crossing points and T denotes the tangency points, then $|X| = \Omega(|T|(\log\log n)^{1/8})$ and conjectured that $|X| = \Omega(|T|\log n)$.

Another line of research studies the number of tangencies between two families each consisting of disjoint curves. This problem was introduced by Pach, Suk and Treml [25] who attributed it to Ben-Dan and Pinchasi [6]. An upper bound of $O(n^{3/2})$ follows from an observation of Ben-Dan and Pinchasi and from the result of Janzer et al. [16], whereas a lower bound of $\Omega(n^{4/3})$ was shown by Keszegh and Pálvölgyi [17] using 2-intersecting curves. For two families of 1-intersecting curves a linear upper bound was proved by Pálvölgyi et nos [3]. For further related results on tangencies among curves and their applications see [5] and [22] and the references therein.

Organization

After introducing some notations and collecting a few useful lemmas in Section 2, we prove the results for arbitrary 1-intersecting curves in Section 3. Our results for x-monotone curves are presented and partially proved in Section 4. Finally, in Section 5 we prove Theorem 9 and discuss its further applications and connections to related graph theoretic results. Proofs that were omitted due to space limitation can be found in the full version of this paper [2].

[4] As mentioned before, their notion of tangency is a bit different and they actually bound the number of tangency points.

2 Notations and Tools

Recall that we assume that every two curves intersect at finitely many points and that no three curves intersect at a single point. It follows from the former assumption that we may also assume that the curves are drawn as polygonal chains. Indeed, given a set of curves $\mathcal{C}$, consider the plane graph G whose vertex set consists of the endpoints and intersection points of the curves in $\mathcal{C}$ and whose edge set consists of subcurves between consecutive vertices along the curves. By Fáry's Theorem [12] this planar graph can be embedded using straight-line edges and the same rotation system. Replacing every curve $c \in \mathcal{C}$ with the polygonal chain that consists of the edges in this embedding that correspond to edges of G that belong to c, we obtain a family of curves $\mathcal{C}'$ such that for every intersection (resp., tangency) point of two curves in $\mathcal{C}$ there is a unique intersection (resp., tangency) point between their corresponding curves in $\mathcal{C}'$, and vice versa.

For a set of curves $\mathcal{C}$ we denote by $G_{\mathcal{C}}$ the *tangency graph* of $\mathcal{C}$, that is, the graph whose vertex set is $\mathcal{C}$ and whose edge set consists of the tangent pairs of curves. By abuse of notation we sometimes interchange curves in $\mathcal{C}$ and their corresponding vertices in $G_{\mathcal{C}}$.

Let c be a curve and let p and q be two points on c. We denote by $c[p,q]$ the subcurve of c between these two points. For another curve c' that intersects c at a single point, we may also write, e.g., $c[c',q]$ instead of $c[c \cap c',q]$. If c is oriented, then we denote by c^- (resp., c^+) its starting (resp., terminating) point following its orientation. We also write $c[-,p]$ and $c[p,+]$ instead of $c[c^-,p]$ and $c[p,c^+]$, respectively. If c is an x-monotone curve and p is a point (not necessarily on c), then with some abuse of notation by writing, e.g., $c[p,+]$, we mean $c[\ell_p \cap c, +\infty)$, where ℓ_p is the vertical line through p. As usual, a round bracket indicates that an endpoint does not belong to the subcurve.

A family of x-monotone curves induces a *generalized trapezoidal partition* of the plane in the following way. From every point p which is an endpoint of a curve or an intersection point of two curves we draw a maximal vertical segment (possibly a ray or a line) that contains p and does not cross any other curve but the ones containing p. This yields a partition of the plane into *generalized trapezoids*.

▶ **Theorem 11** (Cutting Lemma for x-monotone curves [15, Proposition 2.11]). *For every family $\mathcal{C}$ of n x-monotone 1-intersecting curves and any $r > 1$ there is a subset of curves in $\mathcal{C}$ which induces a generalized trapezoidal partition of the plane into $O(r^2)$ generalized trapezoids each intersected by at most n/r curves from $\mathcal{C}$.*

We will also make use of the (asymmetric) Kővári-Sós-Turán Theorem and the recent tight bound of Janzer et al. [16] on the size of intersection-reverse sequences. The latter improves previous bounds by Marcus and Tardos [19] and by Pinchasi and Radoičić [26].

▶ **Theorem 12** (Kővári-Sós-Turán Theorem [18]). *For every $m,n,s,t \geq 1$ if $G = (A \cup B, E)$ is a bipartite graph such that $|A| = m$, $|B| = n$, and there are no s vertices in A and t vertices in B that form a complete bipartite subgraph $K_{s,t}$, then $|E| \leq (s-1)^{1/t} n m^{1-1/t} + (t-1)m$. In particular, $|E| = O_s(n^{2-1/t})$ when $m = n$ and $s \geq t$.*

▶ **Theorem 13** ([16]). *Let $A^1, \ldots, A^n$ be linear orders on some subsets of a set of n symbols such that no three symbols appear in the same order in any two distinct linear orders. Then $\sum_{i=1}^{n} |A^i| = O(n^{3/2})$.*

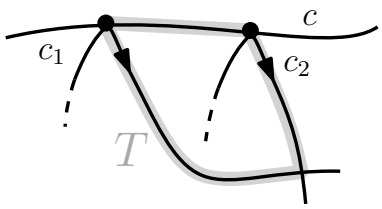


Figure 1 If c_1 and c_2 are grounded at ℓ and have the same tangency type with c, then $c_1[c,+]$ and $c_2[c,+]$ cannot cross.

3 Arbitrary 1-intersecting curves

In this section we consider 1-intersecting curves that are not necessarily x-monotone and prove Theorems 3 and 4. We begin with grounded curves. A family of curves $\mathcal{C}$ is *grounded* at a curve $c \notin \mathcal{C}$ if every curve in $\mathcal{C}$ has an endpoint on c.

▶ **Theorem 14.** *Let A be a family of curves grounded at a curve ℓ_A and let B be a family of curves grounded at a curve ℓ_B, such that all of these curves form a family of 1-intersecting curves, no curve in $A \cup \{\ell_A\}$ intersects ℓ_B and no curve in $B \cup \{\ell_B\}$ intersects ℓ_A. Then the number of tangent pairs $\{c_A \in A, c_B \in B\}$ is $O(|A \cup B|^{3/2})$.*

Proof. We refer to the curves in A as *red* curves and to the curves in B as *blue* curves. If a pair of curves of the same color touch, then we may redraw them locally such that they are disjoint, since we only care about red-blue tangencies. Thus, every pair of curves of the same color are either crossing or disjoint. Orient each curve in A away from ℓ_A and each curve in B away from ℓ_B. Then, there are four types of red-blue tangencies since every curve has a "left" side and a "right" side with respect to its orientation.

▶ **Proposition 15.** *Let c, c_1, c_2 and ℓ be four distinct curves such that: (1) $\{c, c_1, c_2, \ell\}$ is 1-intersecting; (2) c_1 and c_2 are grounded at ℓ and are oriented away from ℓ; (3) c is oriented and touches c_1 and c_2 at tangency points of the same type; and (4) c and ℓ are disjoint. Then $c_1[c,+]$ and $c_2[c,+]$ do not cross.*

Proof. Suppose that $c_1[c,+]$ and $c_2[c,+]$ cross and let T denote the simple closed curve $c[c_1,c_2] \cup c_1[c,c_2] \cup c_2[c,c_1]$, see Figure 1. Then c_1^- and c_2^- lie on different sides T.[5] However, this is impossible since both of these points are on ℓ which cannot cross T. ◀

Returning to the proof of the theorem, we consider each of the four tangency types separately. For a certain type t, we write for each curve $b \in B$ the ordered list of curves from A that touch b at a touching point of the given type. The list, denote it by $l_t(b)$, is ordered according to the order of the corresponding touching points along b.

▶ **Proposition 16.** *There are no three curves $a_1, a_2, a_3 \in A$ and two curves $b_1, b_2 \in B$ such that a_1, a_2, a_3 appear in this order both in $l_t(b_1)$ and in $l_t(b_2)$.*

Proof. Suppose for contradiction that there are such curves. Observe first that it follows from Proposition 15 that $b_1[a_1,+]$ and $b_2[a_1,+]$ do not cross. In particular $b_1[a_1,a_3]$ and $b_2[a_1,a_3]$ do not cross. Next, we claim that at least one pair of the three subcurves $a_i[b_1,b_2]$, $i = 1,2,3$, is crossing. Suppose for contradiction that these curves are pairwise disjoint. Slightly "inflate"

[5] Assuming some arbitrary orientation of T, disregarding the orientations of the other curves.

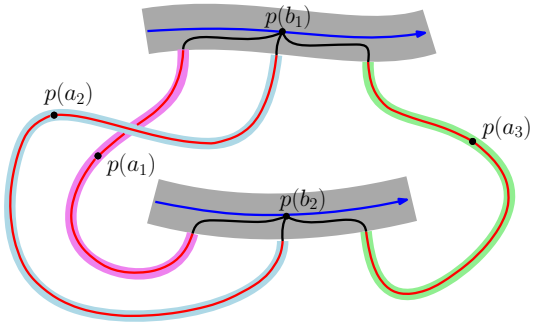


Figure 2 If the subcurves $a_i(b_1, b_2)$ were disjoint, then we could draw a crossing-free copy of $K_{2,3}$.

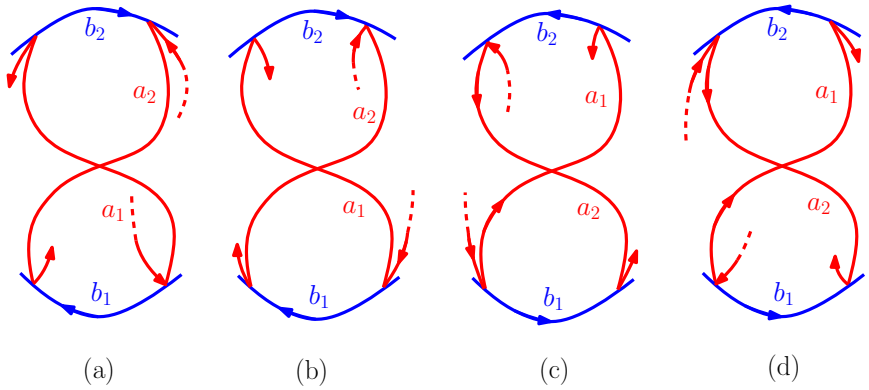


Figure 3 Illustrations for the proof of Proposition 16: $a_1[b_1, +]$ and $a_2[-, b_1]$ are crossing and together with $b_1[a_1, a_2]$ and $b_2[a_1, a_2]$ induce a partition of the plane. a_1^- and a_2^- lie on different regions of this partition for any red-blue tangency type.

b_1 and b_2 and fix a point $p(c)$ on each subcurve $c \in \{a_1(b_1, b_2), a_2(b_1, b_2), a_3(b_1, b_2), b_1(a_1, a_3), b_2(a_1, a_3)\}$. Note that these subcurves are disjoint. Next, draw a crossing-free copy of $K_{2,3}$ whose vertices are these five points and whose edges follow the corresponding subcurves, as suggested by Figure 2. Note that since a_1, a_2, a_3 touch in this order both of b_1 and b_2, the counterclockwise cyclic order of $p(a_1), p(a_2), p(a_3)$ is the same for $p(b_1)$ and $p(b_2)$. However, this contradicts the following lemma of Pinchasi and Radoičić [26].

▶ **Lemma 17** ([26, Lemma 1]). *Let G be a topological graph with no two edges belonging to a 4-cycle that crosses itself an odd number of times. For each vertex v of G let C_v denote the counterclockwise cyclic order of the neighbors of v. Then for each pair of vertices u and v, if there are more than two common neighbors of u and v, then the order of these neighbors in C_v is reversed with respect to their order in C_u.*

For the rest of the proof we will only consider the guaranteed pair of crossing subcurves among $a_i[b_1, b_2]$, $i = 1, 2, 3$, therefore we may suppose without loss of generality that $a_1(b_1, b_2)$ and $a_2(b_1, b_2)$ are crossing. We may also assume that a_1 meets b_1 before meeting b_2. Since $a_1[b_1, +]$ cannot cross $a_2[b_1, +]$ by Proposition 15, it follows that $a_1[b_1, +]$ and $a_2[-, b_1]$ are crossing. Together with $b_1[a_1, a_2]$ and $b_2[a_1, a_2]$ they induce a partition of the plane into three connected regions. For each type of red-blue tangency, the points a_1^- and a_2^- lie on two different regions, see Figure 3 for an illustration. However, this is impossible since both of these points lie on ℓ_A which does not cross any of the curves a_1, a_2, b_1, b_2. ◀

It follows from Proposition 16 and Theorem 13 that $\sum_i |l_t(b_i)| = O(|A \cup B|)^{3/2}$. Therefore, the number of red-blue tangencies is also $O(|A \cup B|)^{3/2}$. ◀

▶ **Corollary 18.** *Let $\mathcal{C}$ be a family of n 1-intersecting curves grounded at a curve ℓ. Then the number of tangencies among $\mathcal{C}$ is $O(n^{3/2})$.*

Proof. Denote by $t(n)$ the maximum number of tangencies among such a family of n curves. We prove that $t(n) = O(n^{3/2})$ by induction on n. Split ℓ into two disjoint subcurves such that $n/2$ curves from $\mathcal{C}$ are grounded at each subcurve. Using Theorem 14 and induction we get the recursive relation $t(n) \leq 2t(n/2) + O(n^{3/2})$ which implies that $t(n) = O(n^{3/2})$. ◀

▶ **Corollary 19.** *Let A be a family of curves grounded at a curve ℓ_A and let B be a family of curves grounded at a curve ℓ_B, such that $A \cup B \cup \{\ell_A, \ell_B\}$ form a family of 1-intersecting curves. Then the number of tangent pairs $\{c_A \in A, c_B \in B\}$ is $O(|A \cup B|^{3/2})$.*

Proof. If ℓ_A and ℓ_B cross at a point p, then we split each of them into two curves at p (such that p belongs to neither of them). Denote these subcurves by $\ell_A^1, \ell_A^2, \ell_B^1$ and ℓ_B^2 and note that each pair of them are disjoint. Every curve in A (resp., B) that crosses ℓ_B (resp., ℓ_A) we cut into two subcurves, such that one of them is grounded at ℓ_A^1 or ℓ_A^2 (or at ℓ_A if it was not cut) and the other is grounded at ℓ_B^1 or ℓ_B^2 (or at ℓ_B if it was not cut). It remains to bound the number of tangencies among curves grounded at ℓ_x^i, for each $x \in \{A, B\}$ and $i \in \{1, 2\}$, and the number of tangencies of pairs of curves c and c' such that c is grounded at ℓ_x^i and c' is grounded at ℓ_y^j for each $x, y \in \{A, B\}$ and $i, j \in \{1, 2\}$ such that $x \neq y$ or $i \neq j$. It follows from Corollary 18 and Theorem 14 that the total number of all of these tangencies is $O(|A \cup B|^{3/2})$. ◀

We can now deduce the main results of this section.

▶ **Theorem 4.** *Every family of n precisely 1-intersecting planar curves admits $O(n^{3/2})$ tangencies.*

Proof. Follows immediately from Corollary 18. ◀

▶ **Theorem 3.** *Every family of n 1-intersecting planar curves admits $O(n^{5/3})$ tangencies.*

Proof. Let $\mathcal{C}$ be a family of n 1-intersecting curves and let $G_{\mathcal{C}}$ be the tangency graph of $\mathcal{C}$. It follows from Corollary 19 that $G_{\mathcal{C}}$ satisfies the f-sparse sub-bineighborhoods property for $f(x) = O(x^{3/2})$. Indeed, let v_0 and v_1 be two adjacent vertices in $G_{\mathcal{C}}$ and let $V_i \subseteq N_{G_{\mathcal{C}}}(v_i) \setminus \{v_{1-i}\}$, for $i = 0, 1$, be two disjoint subsets of their neighborhoods in $G_{\mathcal{C}}$. We wish to bound $t(V_0, V_1)$ – the number of tangent pairs $\{u_0 \in V_0, u_1 \in V_1\}$. For every $i = 0, 1$ and every curve $u \in V_i$, we split u into two subcurves u_i' and u_i'' such that each of them is grounded at v_i. Denote the resulting set of curves by V_1', V_1'', V_2' and V_2'', respectively. Thus, $t(V_0, V_1) = t(V_0', V_1') + t(V_0', V_1'') + t(V_0'', V_1') + t(V_0'', V_1'')$. It follows from Corollary 19 that this sum is upper-bounded by $O(|V_0 \cup V_1)|^{3/2})$. Therefore, by Theorem 9 we have that $|E(G_{\mathcal{C}})| \leq O(n^{5/3})$. ◀

4 Tangencies among x-monotone curves

Theorem 7 is proved in the full version of this paper [2]. Here we give a short summary of the ideas of the proof. First, in the case of exactly 1-intersecting bi-infinite x-monotone curves we prove that the tangency graph cannot contain a cycle, thus has at most $n - 1$ edges. Then we prove a linear bound for a bipartite variant of counting tangencies between two families of bi-infinite x-monotone curves that are separated in the infinity and apply this to get the $O(n \log n)$ bound for the general 1-interesting case.

Next we show the main ideas needed to prove Theorem 5 and 6, the details are again in the full version of this paper [2].

Pach and Sharir [23] studied the following problem: What is the maximum number of pairs of disjoint line-segments that are *vertically visible* among a set of n segments in the plane? We say that two segments are *vertically visible* if there exists a vertical segment that intersects both of them and does not intersect any other segment in the set. The motivation for this problem came from analyzing the maximum size of the events queue in the original implementation of the Bentley-Ottmann line sweeping algorithm for enumerating all the intersections among a set of segments [7].

Pach and Sharir [23] proved that the maximum number of pairs of disjoint vertically visible line-segments within a set of n segments is $O\left(n^{4/3}\left(\log n\right)^{2/3}\right)$ and $\Omega\left(n^{4/3}\right)$. They also mentioned these bounds hold also for x-monotone 1-intersecting curves.

It is easy to see that counting disjoint vertically visible pairs can be reduced to counting touching pairs of x-monotone 1-intersecting curves and vice versa. Indeed, vertically visible curves can become tangent by replacing a very narrow part of one curve by a "spike" that touches the other curve. On the other hand, a pair of touching curves can be easily "detached" and become vertically visible. Therefore, the result of Pach and Sharir [23] implies that the maximum number of tangencies among n x-monotone 1-intersecting curves is $O\left(n^{4/3}\left(\log n\right)^{2/3}\right)$ and $\Omega\left(n^{4/3}\right)$.

We outline the proof of the upper bound of Pach and Sharir [23] and point out the modifications which lead to the better bounds of Theorem 5 and 6. First, the cutting lemma (Theorem 11) is used to partition the plane into generalized trapezoids such that every trapezoid is cut by not too many curves. Then, the number of tangencies within every trapezoid T_i is estimated and finally the sum over all the trapezoids is bounded. Regarding tangencies within T_i, one considers two subsets of curves that cut T_i: "short" curves that have at least one endpoint inside T_i and "long" curves that have no endpoint inside T_i. Therefore, the long curves behave like bi-infinite curves with respect to T_i. Thus, the number of long-long tangencies can be bounded using the $O(n \log n)$ bound of Theorem 7. In order to obtain the tight upper bound of Theorem 6 we use for this case the linear bound for the case of counting tangencies between two families of bi-infinite x-monotone curves that are separated in the infinity, see Lemma 2 in the full version of this paper [2]. The number of short-long tangencies is bounded using Kővári-Sós-Turán Theorem and a technical lemma (see the full version of this paper [2]). Finally, the number of short-short tangencies is bounded in [23] by further subdividing T_i into vertical slabs. However, we observe that one can instead use induction and this simplifies the proof and leads to a better upper bound.

4.1 Bi-infinite k-intersecting x-monotone curves

In light of the second part of Theorem 7 and [5, Theorem 2.4] it is tempting to suggest that a similar statement holds for every fixed k.

▶ **Conjecture 20.** *Every set of n pairwise intersecting bi-infinite x-monotone k-intersecting curves admits $O_k(n)$ tangencies.*

We were unable to settle this conjecture. Still, we mention a simple and weak polynomial upper bound and two observations regarding the conjecture. For the upper bound we use the following observation.

▶ **Observation 21.** *Let r and r' be two bi-infinite x-monotone curves touching from below an x-monotone curve b. Then r and r' cross at a point whose x-coordinate is between $b \cap r$ and $b \cap r'$.*

By Theorem 2 we have $O_k\left(n^{2-\frac{1}{k+3}}\right)$ tangencies among any n k-intersecting curves. One can slightly improve upon this bound in case of bi-infinite x-monotone curves.

▶ **Proposition 22.** *Let $\mathcal{L}$ be a family of k-intersecting bi-infinite x-monotone curves. Then $\mathcal{L}$ admits $O_k\left(n^{2-\frac{1}{k+1}}\right)$ tangencies.*

Proof. Note that the curves in $\mathcal{L}$ are not necessarily pairwise intersecting. Consider the tangency graph $G_{\mathcal{L}}$ and flip a fair coin for every curve $c \in \mathcal{L}$. If the outcome is "heads", delete all the edges that correspond to tangencies in which c touches another curve from below. Otherwise, if the outcome is "tails" do the same for tangencies in which c touches a curve from above. Clearly, every edge in $G_{\mathcal{L}}$ survives with probability $1/4$. Therefore, there is a bipartite subgraph $(B \cup R, E)$ with at least $|E(G_{\mathcal{L}})|/4$ edges in which every curve that corresponds to a vertex $r \in R$ touches from below the curves corresponding to its neighbors in B. It suffices to bound the size of this subgraph.

We claim that this subgraph does not contain K_{k+1,c_k} as a subgraph, for some large enough c_k to be determined later. Indeed, suppose without loss of generality that there is a subset of c_k curves $R' = \{r_1, r_2, \ldots, r_{c_k}\} \subseteq R$ each of which touches from below each of the $k+1$ curves in a subset $B' = \{b_1, b_2, \ldots, b_{k+1}\} \subseteq B$. Let ℓ be the lower envelope of the curves in B'. Then ℓ consists of at most $\lambda_{k+2}(k+1)$ curve-segments, where $\lambda_s(n)$ denotes that maximum length of an (n, s)-Davenport-Schinzel sequence [27].[6] Note that each curve $r' \in R'$ touches $k+1$ of these curve-segments (which belong to $k+1$ different curves in B'). Therefore, there are at most $\binom{\lambda_{k+2}(k+1)}{k+1}$ possibilities for the $(k+1)$-subset of curve-segments of ℓ that a curve in R' touches. Setting $c_k = 1 + \binom{\lambda_{k+2}(k+1)}{k+1}$ implies that there are two curves $r', r'' \in R'$ that touch the same $k+1$ curve-segments of ℓ. Consider one of these curve-segments s. Then it follows from Observation 21 that r' and r'' cross at a point whose x-coordinate is between $s \cap r'$ and $s \cap r''$. Since the $k+1$ curve-segments that r' and r'' touch have distinct x-projections, it follows that r' and r'' cross at least $k+1$ times, which is impossible. Thus, $(B \cup R, E)$ is a K_{k+1,c_k}-free graph and it follows from Kővári-Sós-Turán Theorem that it has at most $O_k\left(n^{2-\frac{1}{k+1}}\right)$ edges. ◀

The next observation says that if Conjecture 20 is true, then the same statement holds without requiring the curves to be bi-infinite.

▶ **Proposition 23.** *Suppose that Conjecture 20 holds for some $k > 2$. Then every set of n pairwise intersecting x-monotone $(k-2)$-intersecting curves admits $O_k(n)$ tangencies.*

Proof. Let $\mathcal{L}$ be a set of curves as in the proposition. We assume without loss of generality that no two endpoints of two curves share the same x-coordinate. There are four types of tangencies: the curve touching from above starts before/after the curve touching from below and ends before/after it. We consider each type separately and observe that the set of curves touching from above and the set of curves touching from below are disjoint sets. Call the curves in the first set *red* and the curves in the other set *blue*. Next, we extend each red curve

[6] A sequence of letters over an n-element alphabet is an (n, s)-*Davenport-Schinzel sequence* if it does not contain two consecutive identical letters and no alternating sub-sequence of two letters whose length is $s+2$. In fact a trivial bound of $O(k^3)$ on the size of the lower envelope also suffices for our purpose.

into a bi-infinite curve by shooting a very steep ray leftwards (resp., rightwards) and upwards at its left (resp., right) endpoint. Similarly, we extend each blue curve into a bi-infinite curve by shooting a very steep ray leftwards (resp., rightwards) and downwards at its left (resp., right) endpoint. The rays going leftward and upwards and the rays going rightwards and downwards have opposite slopes. Similarly, the rays going leftward and downwards and the rays going rightwards and upwards have opposite slopes. Note that this introduces at most two new intersection points between two curves while for touching curves (of the considered type) no new intersection point is introduced. ◀

According to the following observation for an even k we may assume that every non-tangent pair of curve intersects at exactly k points.

▶ **Proposition 24.** *Let $\mathcal{L}$ be a set of n bi-infinite x-monotone curves such that every pair of curves in $\mathcal{L}$ intersect at at least one and at most k points for some even k. Then there is a set $\mathcal{L}'$ of n bi-infinite x-monotone curves such that every pair of curves in $\mathcal{L}'$ is either touching at a single point or crossing at exactly k points and the number of touching pairs in $\mathcal{L}$ and $\mathcal{L}'$ is the same.*

Proof. We may assume without loss of generality that all the curves are parallel to the x-axis at $-\infty$ and $+\infty$ and denote them by $1, 2, \ldots, n$ according to their order from top to bottom at $-\infty$. Let $f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ denote their order at $+\infty$, that is, $f(i)$ is the location of i in the order of the curves from top to bottom at $+\infty$.

If f is the identity function, then every pair of curves either touch or cross at an even number of points. In such a case, every two curves that cross at $k' < k$ points, can be redrawn very close to one of their crossing points such that they cross at $k - k'$ new crossing points. This way we get exactly k crossing points between every pair of non-touching curves. Otherwise, if f is not the identity function, then we can redraw two curves that cross an odd number of times such that they cross an even number of times and repeat this process until f becomes the identity function. Indeed, let i and j be two curves such that $i < j$ and $f(i) > f(j)$. If there are several such pairs we choose a "closest" one, that is, one which minimizes the difference $f(i) - f(j)$. Note that there is no curve l such that $f(i) > f(l) > f(j)$, for otherwise we must have both $l < i$ and $j < l$ (or else (i, l) or (l, j) would be a "reversed" pair closer than (i, j)) which is impossible since $i < j$. Therefore, $f(i) = f(j) + 1$ and it follows that we can redraw i and j such that they cross at an additional point. ◀

5 An Erdős-Simonovits-type theorem

In this section we prove Theorems 9 and 10 and discuss some of their applications and connections to other graph theoretic results. For a graph $G = (V, E)$ we denote by $\overline{d}(G) = 2|E|/|V|$ the average degree in G. The subgraph induced by a subset of vertices $U \subseteq V$ is denoted by $G[U]$. For a vertex $v \in V$ we denote by $N_G(v)$ (or simply $N(v)$ when the context is clear) the neighbors of v in G. The (*open*) *bineighborhood* of two vertices v and u is equal to $(N_G(v) \cup N_G(u)) \setminus \{u, v\}$. Assume henceforth that G is bipartite and recall that for a nonnegative and nondecreasing function $f : \mathbb{R} \to \mathbb{R}$ we say that two vertices u and v have *f-sparse sub-bineighborhoods* if for every two disjoint subsets $U \subseteq N_G(u) \setminus \{v\}$ and $V \subseteq N_G(v) \setminus \{u\}$ it holds that $|E(G(U, V))| \leq f(|U \cup V|)$, where $G(U, V)$ is the bipartite subgraph $(U \cup V, \{(u, v) \in E(G) \mid u \in U \text{ and } v \in V\})$. Theorem 9 will follow from the next result.

▶ **Theorem 25.** *Let $f : \mathbb{R} \to \mathbb{R}$ be a nonnegative and nondecreasing function and let $G = (A \cup B, E)$ be an n-vertex bipartite graph with $\overline{d}(G) \geq 16$.*

(i) *If every pair of vertices in G has f-sparse sub-bineighborhoods, then $(\overline{d}(G))^3 \leq 2^{11} n f(2\overline{d}(G))$.*

(ii) *If every pair of adjacent vertices has f-sparse sub-bineighborhoods and $|E| \geq 25n^{3/2}$, then $(\overline{d}(G))^3 \leq 2^{20} n f(2\overline{d}(G))$.*

Proof. Set $m = |E|$ and let $d = \overline{d}(G) = 2m/n$ be the average degree of G. We first replace G with a graph G' in which the degree of every vertex is roughly the average degree (while maintaining the f-sparseness property): Go over the vertices of G in an arbitrary order and replace each vertex v such that $\deg(v) > d$ by $\lfloor \deg(v)/d \rfloor$ vertices of degree d and possibly one vertex of a smaller degree. The neighbors of v are connected to the vertices that replace v in an arbitrary manner which satisfies this property. Observe that the resulting graph still has f-sparse sub-bineighborhoods for every pair of (adjacent) vertices, since each sub-neighborhood of a vertex in the new graph cannot contain two vertices that correspond to the same original vertex. Thus, every sub-neighborhood in the new graph corresponds to a sub-neighborhood in G. Furthermore, since $d = 2m/n$, the number of vertices in the new graph is at most $\sum_{v \in A \cup B} (\lfloor \deg_G(v)/d \rfloor + 1) \leq 2n$. Because the number of edges remains the same, the average degree of this graph is at least $d/2$.

Next, we repeatedly remove vertices of degree smaller than $d/8$ until no such vertices remain. We claim that at least $m/2$ edges remain after this step. Indeed, suppose for contradiction that less than $m/2$ edges remain and denote by x the number of vertices that were removed. Then at most $xd/8$ edges were removed, therefore $xd/8 > m/2 = 2nd/8$ which is impossible since $x \leq 2n$. Denote by $G' = (A' \cup B', E')$ the resulting graph and note that $|A' \cup B'| \leq 2n$, $|E'| \geq m/2$ and the degree of every vertex in G' is at least $d/8$ and at most d. Observe also that $|A' \cup B'| \geq 2|E'|/d \geq m/d = n/2$. Furthermore, G' still has the f-sparse sub-bineighborhoods property, since this property is maintained when deleting a vertex.

Considering (i), for each pair of vertices $a \in A'$ and $b \in B'$ the number of edges in the subgraph of G' induced by the union of their open bineighborhoods is at most $f(2d)$. Therefore we have $\sum_{a \in A', b \in B'} |E(G'[(N_{G'}(a) \setminus \{b\}) \cup (N_{G'}(b) \setminus \{a\}])| \leq \binom{2n}{2} f(2d) \leq 2n^2 f(2d)$. Note that every edge $(a, b) \in E'$ is counted in this sum for every pair $a' \in A'$ and $b' \in B'$ such that $a \in N_{G'}(b') \setminus \{a'\}$ and $b \in N_{G'}(a') \setminus \{b'\}$. Since a (resp., b) has at least $\frac{d}{8} - 1 \geq \frac{d}{16}$ neighbors different from b (resp., a), it follows that (a, b) is counted at least $d^2/256$ times in the above sum (here we used the assumption $d \geq 16$). Therefore, $nd/2 = m = |E| \leq 2|E'| \leq 2^{10} n^2 f(2d)/d^2$ and thus $d^3 \leq 2^{11} n f(2d)$.

Considering (ii), we have $d \geq 50n^{1/2}$ since $m \geq 25n^{3/2}$. Assume w.l.o.g. that $|B'| \geq |A'|$ and therefore $|B'| \geq n/4$ and $|A'| \leq n$. Let $\#K_{2,1}$ be the number of $K_{2,1}$'s in G' with two vertices in A' and let $\#K_{2,2}$ be the number of $K_{2,2}$'s in G'. Then,

$$\#K_{2,2} = \frac{1}{4} \sum_{a \in A', b \in B', (a,b) \in E'} |E(G'[(N_{G'}(a) \setminus \{b\}) \cup (N_{G'}(b) \setminus \{a\}])| \leq \frac{m}{4} f(2d) = \frac{dn}{8} f(2d).$$

On the other hand, denoting $s_{a,a'} = |N_{G'}(a) \cap N_{G'}(a')|$ for two distinct vertices $a, a' \in A'$ we have $\#K_{2,1} = \sum_{a \neq a' \in A'} s_{a,a'}$ and:

$$\begin{aligned} \#K_{2,2} &= \sum_{a \neq a' \in A'} \binom{s_{a,a'}}{2} = \frac{1}{2} \sum_{a \neq a' \in A'} (s_{a,a'}^2 - s_{a,a'}) \\ &\geq \frac{(\#K_{2,1})^2}{2\binom{|A'|}{2}} - \frac{\#K_{2,1}}{2} \geq \frac{(\#K_{2,1})^2}{2\binom{n}{2}} - \frac{\#K_{2,1}}{2}, \end{aligned}$$

where the second to last inequality follows from Cauchy's inequality. For $\#K_{2,1}$ we have the following lower bound:

$$\#K_{2,1} = \sum_{a \neq a' \in A'} s_{a,a'} = \sum_{b \in B'} \binom{\deg(b)}{2} \geq \frac{n}{4}\binom{d/8}{2} \geq nd^2/2^{11} \geq n^2,$$

where we used double-counting by the vertex from B' for the $K_{2,1}$'s and the inequalities $|B'| \geq n/4$ and $d \geq 50n^{1/2}$. Combining the inequalities above we get:

$$\begin{aligned}\frac{dnf(2d)}{8} &\geq \#K_{2,2} \geq \frac{(\#K_{2,1})^2}{n^2} - \frac{\#K_{2,1}}{2} \geq \frac{(\#K_{2,1})^2}{n^2} - \frac{(\#K_{2,1})^2}{2n^2} = \frac{(\#K_{2,1})^2}{2n^2} \\ &\geq \frac{n^2 d^4}{2^{22} \cdot 2n^2} = \frac{d^4}{2^{23}}.\end{aligned}$$

This implies that $d^3 \leq 2^{20} nf(2d)$, as claimed. ◀

Using Theorem 25 we can now prove Theorem 9 which we restate for convenience.

▶ **Theorem 9.** *Let G be a graph and let $f(n) = O(n^{2-\alpha})$ for some constant $0 \leq \alpha < 2$. If every pair of vertices in G has f-sparse sub-bineighborhoods, then $|E(G)| = O\left(|V(G)|^{2-\frac{\alpha}{\alpha+1}}\right)$. Moreover, if $\alpha \leq 1$, then it is enough that every pair of adjacent vertices has f-sparse sub-bineighborhoods for this bound to hold.*

Proof. Recall that every graph contains a bipartite subgraph whose size is at least half of the size of the graph. Let $G' = (A' \cup B', E')$ be such a bipartite subgraph of G, where $|E'| \geq |E(G)|/2$, and let $n = |V(G)|$ and $d' = \overline{d}(G')$. If every pair of vertices in G has f-sparse sub-bineighborhoods, then the same holds for G' and it follows from Theorem 25 that $d'^3 \leq 2^{11} nf(2d') = O(nd'^{2-\alpha})$. Thus, $d' \leq O(n^{1/(\alpha+1)})$ and the upper bound follows.

If $0 < \alpha \leq 1$ and every pair of adjacent vertices has f-sparse sub-bineighborhoods, then we may assume that $|E'| \geq 25n^{3/2}$ for otherwise the theorem trivially holds. Therefore, by Theorem 25 we have $d'^3 \leq 2^{20} nf(2d') = O(nd'^{2-\alpha})$ and $d' \leq O(n^{1/(\alpha+1)})$ as before. ◀

For the lower bound we show that an appropriate random bipartite graph meets the criteria with positive probability, the proof can be found in the full version of this paper [2].

Remarks

- The proof of Theorem 9 is quite similar to the proof of Theorem 8 in [10]. In both cases, one first makes the graph close to being regular and then uses double-counting, however, in the case of Theorem 8 the first part requires more effort. Chronologically, we first proved the first part of Theorem 9 which was needed for the proof of Theorem 3, only later did we learn about Theorem 8 and used it to prove the second part of Theorem 9.
- It would be interesting to determine whether Theorem 9 holds when one assumes sparsity of the subgraph induced by the bineighborhood of every pair of (adjacent) vertices instead of assuming sparsity of subgraphs induced by each of their sub-bineighborhoods. We note that the first part of the proof of Theorem 9 does not go through in such a case. Also, f-sparse bineighborhoods do not imply f-sparse sub-bineighborhoods.
- Theorem 9 might find other applications, e.g., in other scenarios where it can be combined with Theorem 13. Here is one example. For a graph H let $\mathrm{ex}_{\mathrm{cr}}(n, H)$ be the maximum number of edges in a topological graph on n vertices with no self-intersecting copy H. Note that $\mathrm{ex}(n, H) \leq \mathrm{ex}_{\mathrm{cr}}(n, H)$ and if H is non-planar then $\mathrm{ex}(n, H) = \mathrm{ex}_{\mathrm{cr}}(n, H)$. In [16] Theorem 13 is used to show that $\mathrm{ex}_{\mathrm{cr}}(n, C_4) = O(n^{3/2})$ (and this is tight). Using Theorem 9 this immediately implies:

▶ **Corollary 26.** *$ex_{cr}(K_{2,3}) = O(n^{5/3})$ and $ex_{cr}(n, K_{3,3}^{-}) = O(n^{5/3})$ where $K_{3,3}^{-}$ is the complete bipartite graph on $3+3$ vertices minus one edge.*

References

1 Eyal Ackerman, Gábor Damásdi, Balázs Keszegh, Rom Pinchasi, and Rebeka Raffay. The Maximum Number of Digons Formed by Pairwise Intersecting Pseudocircles. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:14, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2025.2.

2 Eyal Ackerman and Balázs Keszegh. On the maximum number of tangencies among 1-intersecting curves, 2026. arXiv:2603.11885.

3 Eyal Ackerman, Balázs Keszegh, and Dömötör Pálvölgyi. On tangencies among planar curves with an application to coloring L-shapes. *European Journal of Combinatorics*, 121:103837, 2024. doi:10.1016/j.ejc.2023.103837.

4 Eyal Ackerman and Balázs Keszegh. On the number of tangencies among 1-intersecting x-monotone curves. *European Journal of Combinatorics*, 118:103929, 2024. doi:10.1016/j.ejc.2024.103929.

5 Pankaj K. Agarwal, Eran Nevo, János Pach, Rom Pinchasi, Micha Sharir, and Shakhar Smorodinsky. Lenses in arrangements of pseudo-circles and their applications. *J. ACM*, 51(2):139–186, 2004. doi:10.1145/972639.972641.

6 Itay Ben-Dan and Rom Pinchasi. personal communication.

7 Jon Bentley and Thomas Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28(9):643–647, 1979. doi:10.1109/TC.1979.1675432.

8 Peter Brass, William O. J. Moser, and János Pach. *Research problems in discrete geometry*. Springer, 2005.

9 Jordan S. Ellenberg, Jozsef Solymosi, and Joshua Zahl. New bounds on curve tangencies and orthogonalities. *Discrete Analysis*, November 4 2016. doi:10.19086/da.990.

10 P. Erdős and M. Simonovits. Some extremal problems in graph theory. In P. Erdős, A. Rényi, and V. T. Sós, editors, *Combinatorial Theory and its Applications, I*, volume 4 of *Colloquia Mathematica Societatis János Bolyai*, pages 377–390. North-Holland, Amsterdam, 1970. Proc. Colloq., Balatonfüred (Hungary), 1969. URL: https://users.renyi.hu/~miki/1970-22ErdSimCube.pdf.

11 Pál Erdős and Branko Grünbaum. Osculation vertices in arrangements of curves. *Geometriae Dedicata*, 1(4):322–333, 1973. doi:10.1007/BF00147765.

12 István Fáry. On straight line-representations of planar graphs. *Acta Scientiarum Mathematicarum*, 12(182-192):229, 1948.

13 Branko Grünbaum. *Arrangements and spreads*. Conference Board of the Mathematical Sciences Regional Conference Series in Mathematics, No. 10. American Mathematical Society, Providence, R.I., 1972.

14 Péter Györgyi, Bálint Hujter, and Sándor Kisfaludi-Bak. On the number of touching pairs in a set of planar curves. *Computational Geometry*, 67:29–37, 2018. doi:10.1016/j.comgeo.2017.10.004.

15 Sariel Har-Peled. Constructing planar cuttings in theory and practice. *SIAM Journal on Computing*, 29(6):2016–2039, 2000. doi:10.1137/S0097539799350232.

16 Barnabás Janzer, Oliver Janzer, Abhishek Methuku, and Gábor Tardos. Tight bounds for intersection-reverse sequences, edge-ordered graphs and applications. *Journal of the London Mathematical Society*, 112(4), October 2025. doi:10.1112/jlms.70324.

17 Balázs Keszegh and Dömötör Pálvölgyi. The number of tangencies between two families of curves. *Combinatorica*, 43(5):939–952, October 2023. doi:10.1007/s00493-023-00041-8.

18 T. Kővári, V. Sós, and P. Turán. On a problem of K. Zarankiewicz. *Colloquium Mathematicae*, 3(1):50–57, 1954. URL: http://eudml.org/doc/210011.

19 Adam Marcus and Gábor Tardos. Intersection reverse sequences and geometric applications. *J. Comb. Theory, Ser. A*, 113(4):675–691, 2006. doi:10.1016/j.jcta.2005.07.002.

20 János Pach. personal communication.

21 János Pach and Pankaj K. Agarwal. *Combinatorial Geometry*, chapter 11, pages 177–178. John Wiley and Sons Ltd, 1995. doi:10.1002/9781118033203.ch11.

22 János Pach, Natan Rubin, and Gábor Tardos. A crossing lemma for Jordan curves. *Advances in Mathematics*, 331:908–940, 2018. doi:10.1016/j.aim.2018.03.015.

23 János Pach and Micha Sharir. On vertical visibility in arrangements of segments and the queue size in the Bentley-Ottmann line sweeping algorithm. *SIAM Journal on Computing*, 20(3):460–470, 1991. doi:10.1137/0220029.

24 János Pach and Micha Sharir. *Combinatorial Geometry and Its Algorithmic Applications: The Alcalá Lectures*, volume 152 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2009. doi:10.1090/surv/152.

25 János Pach, Andrew Suk, and Miroslav Treml. Tangencies between families of disjoint regions in the plane. *Comput. Geom.*, 45(3):131–138, 2012. doi:10.1016/j.comgeo.2011.10.002.

26 Rom Pinchasi and Radoš Radoičić. On the number of edges in a topological graph with no self-intersecting cycle of length 4. In János Pach, editor, *Towards a Theory of Geometric Graphs*, volume 342 of *Contemporary Mathematics*, pages 233–234. American Mathematical Society, 2004.

27 Micha Sharir and Pankaj K. Agarwal. *Davenport–Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.

Lower Bounding the Gromov–Hausdorff Distance in Metric Graphs

Henry Adams
University of Florida, Gainesville, FL, USA

Sushovan Majhi
George Washington University, Washington, DC, USA

Fedor Manin
University of Toronto, Canada

Žiga Virk
University of Ljubljana, Slovenia
Institute IMFM, Ljubljana, Slovenia

Nicolò Zava
Institute of Science and Technology, Klosterneuburg, Austria

Abstract

Let G be a finite, connected metric graph and let X be a subset of G. If X is sufficiently dense in G, we show that the Gromov–Hausdorff distance matches the Hausdorff distance, namely $d_{\mathrm{GH}}(G, X) = d_{\mathrm{H}}(G, X)$. When the metric graph is the circle $G = S^1$ with circumference 2π, a recent study established the equality $d_{\mathrm{GH}}(S^1, X) = d_{\mathrm{H}}(S^1, X)$ whenever $d_{\mathrm{GH}}(S^1, X) < \frac{\pi}{6}$. Our results relax this hypothesis to $d_{\mathrm{GH}}(S^1, X) < \frac{\pi}{3}$, and furthermore, we show that the constant $\frac{\pi}{3}$ is the best possible. We lower bound the Gromov–Hausdorff distance $d_{\mathrm{GH}}(G, X)$ by the Hausdorff distance $d_{\mathrm{H}}(G, X)$ via a simple topological obstruction: the existence of a possibly discontinuous function $f\colon G \to X$ with too small distortion contradicts the connectedness of G.

2012 ACM Subject Classification Mathematics of computing → Topology; Mathematics of computing → Graph theory

Keywords and phrases Gromov–Hausdorff distance, distortion, connectedness, Borsuk–Ulam theorem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.3

Related Version *Full Version*: `https://arxiv.org/abs/2411.09182` [3]

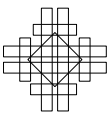
Funding *Henry Adams*: Simons Foundation Travel Support for Mathematicians.
Žiga Virk: Slovene research agency grant P1-0292.
Nicolò Zava: FWF Grant, Project number I4245-N35.

1 Introduction

The Gromov–Hausdorff distance between two abstract metric spaces (X, d_X) and (Y, d_Y), denoted $d_{\mathrm{GH}}(X, Y)$, provides a dissimilarity measure quantifying how far the two metric spaces are from being isometric [13, 14, 24]. In the past two decades, this distance has found applications in topological data analysis (TDA) as a theoretical framework for shape and dataset comparison [21], which motivated the study of its quantitative aspects. However, precise computations of the Gromov–Hausdorff distance between even simple spaces are mostly unknown, with certain exceptions such as the Gromov–Hausdorff distance between a line segment and a circle [16] and between spheres of certain dimensions [19, 1, 15]. Even approximating the Gromov–Hausdorff distance by a factor of 3 in the case of trees with unit-edge length is known to be NP-hard [4, 23]. A better approximation factor can only be achieved in very special cases; for example, [20] provides a $\frac{5}{4}$-approximation scheme if



© Henry Adams, Sushovan Majhi, Fedor Manin, Žiga Virk, and Nicolò Zava;
licensed under Creative Commons License CC-BY 4.0



X and Y are finite subsets of the real line. Developing tools and pipelines to estimate the Gromov–Hausdorff distance is a central research direction in the applied topology community. To estimate the Gromov–Hausdorff distance between two metric spaces X, Y, one may:
(A) Find nice samples $X' \subseteq X$ and $Y' \subseteq Y$ approximating the geometry of X and Y, and
(B) Efficiently bound the Gromov–Hausdorff distance between the subsets X' and Y'.
Indeed, when X and Y are infinite continuous objects – e.g., Riemannian manifolds, metric graphs – and one wants to use computational machinery, it is often essential to resort to finite subsets approximating the geometry of the original spaces.

We reformulate the first task **(A)** as follows: how dense does the sample X' have to be in X to capture the geometry of the ambient space X? To systematize this question, we use the Hausdorff distance as a measure of density and ask if we can provide a threshold ε and constant $0 < C \leq 1$ such that $d_{\mathrm{GH}}(X, X') \geq C \cdot d_{\mathrm{H}}(X, X')$ whenever $d_{\mathrm{H}}(X, X') < \varepsilon$. In particular, $d_{\mathrm{GH}}(X, X') = d_{\mathrm{H}}(X, X')$ if $C = 1$. In [2], the authors show that when X is a manifold and $d_{\mathrm{GH}}(X, X') < \varepsilon$ (which is implied by $d_{\mathrm{H}}(X, X') < \varepsilon$) for some constant ε, then $d_{\mathrm{GH}}(X', X) \geq C \cdot d_{\mathrm{H}}(X, X')$ for some $\frac{1}{2} \leq C \leq 1$. Our study continues that investigation and proves improved results when X is a metric graph, hence relaxing the manifold assumption.

A fruitful approach to provide bounds on the Gromov–Hausdorff distance in task **(B)** is to use stable metric invariants. One associates each metric space X with a value $\psi(X)$ in a simpler metric space so that $\psi(X)$ and $\psi(Y)$ are close whenever X and Y are. An immediate example is the diameter; we have $d_{\mathrm{GH}}(X, Y) \geq \frac{1}{2}|\mathrm{diam}(X) - \mathrm{diam}(Y)|$. For further stable metric invariants, we refer the reader to curvature sets [22], to Vietoris–Rips persistence diagrams [11], and to hierarchical clustering [10]. In [25], dimension theory is used to limit the precision of metric invariants with values in Hilbert spaces.

One of the precursors of algebraic topology was Brouwer's proof of *invariance of dimension*, namely that $\mathbb{R}^m$ is not homeomorphic to $\mathbb{R}^n$ for $n > m$. Proving $\mathbb{R}^1$ is not homeomorphic to $\mathbb{R}^n$ for $n > 1$ relies only on connectedness: removing a single point from $\mathbb{R}^1$ leaves a disconnected space. To prove invariance of dimension for $n > m \geq 2$, Brouwer relied on concepts related to the *degree* of a map between spheres [8, 7], which is often now formalized using higher-dimensional homology groups. The topological obstructions to small Gromov–Hausdorff distances in [2] relied on homological tools, namely the fundamental class of a manifold. One of the key contributions of our paper is to go "back in time" and produce strong lower bounds on Gromov–Hausdorff distances using simpler topological tools: we show that small Gromov–Hausdorff distances would contradict the connectedness of a space.

Overview of our results

Let G be a connected, finite metric graph, and let $X \subseteq G$ be a subset. We show how to lower bound the Gromov–Hausdorff distance $d_{\mathrm{GH}}(G, X)$ in terms of the Hausdorff distance $d_{\mathrm{H}}(G, X)$, achieving $d_{\mathrm{GH}}(G, X) = d_{\mathrm{H}}(G, X)$. A unifying theme in our proofs is that we show that a possibly discontinuous function $f\colon G \to X$ induces a continuous and non-surjective map $\overline{f}\colon G \to G$. We then use non-surjectivity (and related properties) to lower bound the distortion of $\overline{f}$ and f, hence lower bounding $d_{\mathrm{GH}}(G, X)$. The simplest metric graph is a tree T. Let ∂T denote the leaves of T, and let $\vec{d}_{\mathrm{H}}(\partial T, X)$ denote the directed Hausdorff distance from ∂T to the sample X (see Section 2). We show:

Theorem 8. Let T be a finite metric tree and let $X \subseteq T$. If $d_{\mathrm{H}}(T, X) > \vec{d}_{\mathrm{H}}(\partial T, X)$, then $d_{\mathrm{GH}}(T, X) = d_{\mathrm{H}}(T, X)$. The interpretation is as follows: if the biggest gaps in the sampling $X \subseteq T$ are not near the leaves of T (which by Proposition 6 is typical behavior), then $d_{\mathrm{GH}}(T, X) = d_{\mathrm{H}}(T, X)$. Some control on the leaf vertices is needed:

Theorem 5. For any $\varepsilon > 0$ there exists a finite metric tree T and a closed subset $X \subseteq T$ such that $d_{\mathrm{GH}}(T, X) \leq \varepsilon \cdot d_{\mathrm{H}}(T, X)$.

When $G = S^1$ is the circle of circumference 2π, [2] shows that if $d_{\mathrm{GH}}(S^1, X) < \frac{\pi}{6}$, then $d_{\mathrm{GH}}(S^1, X) = d_{\mathrm{H}}(S^1, X)$. The following theorem improves the constant $\frac{\pi}{6}$ to $\frac{\pi}{3}$.

Theorem 11. For any subset $X \subseteq S^1$, we have $d_{\mathrm{GH}}(S^1, X) \geq \min\left\{d_{\mathrm{H}}(S^1, X),\ \frac{\pi}{3}\right\}$.

In **Theorem 12** we show that this constant $\frac{\pi}{3}$ is the best possible, namely that for any $\varepsilon \in \left(0, \frac{\pi}{6}\right)$, there exists a nonempty compact subset $X \subseteq S^1$ with $d_{\mathrm{H}}(S^1, X) = \frac{\pi}{3} + \varepsilon$ and $d_{\mathrm{GH}}(S^1, X) = \frac{\pi}{3} < d_{\mathrm{H}}(S^1, X)$. We generalize the above results for the circle and trees to general metric graphs. Let G be a finite metric graph with set of leaf vertices ∂G. We prove that if X is dense enough relative to $e(G_0)$, which is the smallest edge length between vertices in the core G_0 of G (see Section 2), then the Gromov–Hausdorff distance matches the Hausdorff distance.

Theorem 23. Let G be a finite connected metric graph and let $X \subseteq G$. If $\partial G \neq \emptyset$, suppose $d_{\mathrm{H}}(G, X) > \vec{d}_{\mathrm{H}}(\partial G, X)$. If $d_{\mathrm{GH}}(G, X) < \frac{e(G_0)}{16}$, then $d_{\mathrm{GH}}(G, X) = d_{\mathrm{H}}(G, X)$. Theorems 11 and 23 are obtained using low-dimensional versions of the Borsuk–Ulam theorem for maps from the circle into $\mathbb{R}$ or into a tree, which are proven using connectedness. Our results have corollaries allowing us to lower bound the Gromov–Hausdorff distance between two subsets $X, Y \subseteq G$, using the triangle inequality. That is, for two sufficiently dense subsets $X, Y \subseteq G$, we provide new lower bounds on $d_{\mathrm{GH}}(X, Y)$ in terms of the Hausdorff distance $d_{\mathrm{H}}(X, Y)$, which is $\mathcal{O}(n^2)$-time computable if the subsets have at most n points. We demonstrate such a corollary explicitly in the case of metric trees in Section 4.

Organization of the paper

We begin in Section 2 with background. In Section 3, we show that some control over the boundary of the metric graph is necessary. We begin with metric trees in Section 4, proceed to the circle in Section 5, and consider general metric graphs in Section 6.

2 Background and notation

We refer the reader to [9, 6] for more information on metric spaces, metric graphs, the Hausdorff distance, and the Gromov–Hausdorff distance.

2.1 Metric spaces

Let (Z, d) be a metric space. For $z \in Z$, we let $B(z; r) = \{z' \in Z \mid d(z, z') < r\}$ denote the open metric ball of radius r about z. For any $r \geq 0$ and $A \subseteq Z$, we denote the *r-thickening* of A in Z, namely, the union of metric balls of radius r centered at the points of A, by

$$A^r = \cup_{a \in A} B(a; r) = \left\{z \in Z \mid \inf_{a \in A} d(a, z) < r\right\}.$$

The *diameter* of a subset $A \subseteq Z$ is the supremum of all distances between pairs of points of A. More formally, the diameter is defined as $\mathrm{diam}(A) = \sup\{d(a, a') \mid a, a' \in A\}$.

2.2 Metric graphs

To define metric graphs, we follow Definitions 3.1.12 (quotient metric) and 3.2.9 (metric graph) of [9]. A *metric segment* is a metric space isometric to a real line segment $[0, l]$ for $l \geq 0$. Our metric graphs will be connected and have a finite number of vertices and edges.

▶ **Definition 1** (Metric Graph). *A* finite metric graph (G, d) *is the metric space obtained by gluing a finite collection E of metric segments along their boundaries to a finite collection V of vertices, and equipping the result $\left(\bigsqcup_{e \in E} e\right) / \sim$ with the quotient metric. We assume this equivalence relation results in a connected space.*

The metric segments $e \in E$ are the *edges* and the equivalence classes of their endpoints in V are the *vertices* of G. Since G is connected, we have $d(y, y') < \infty$ for all $y, y' \in G$. Since E is finite, the quotient semi-metric [9, Definition 3.1.12] is in fact a metric.[1] For an edge $e \in E$, its length $|e|$ is the length of the corresponding metric segment. Note that the length can be different from the distance between the endpoints of e according to the metric d. Extending the definition of length for a continuous path $\gamma\colon [a, b] \to G$, we define the length $|\gamma|$ as the sum of the (full or partial) lengths of the edges that γ traverses. For a vertex v of a graph, its *degree*, denoted $deg(v)$, is the number of edges incident to v. A self-loop at v contributes 2 to the degree of v. As an example, the circle S^1 can be given a metric graph structure with a single vertex v of degree 2 at the north pole, along with a self-loop. We denote the set of leaf vertices, i.e. the vertices of degree one, by ∂G. A metric graph G is therefore endowed with two layers of information: a metric structure turning it into a length space [9, Definition 2.1.6] and a combinatorial structure (V, E) as an abstract graph. Using the above definition, we are also allowing G to have single-edge cycles and multiple edges between a pair of vertices. In this paper, we only consider path connected metric graphs (G, d) with finitely many edges. Therefore, the metric graphs we consider are compact, and since a compact length space is geodesic, they are also geodesic (meaning for every $x, y \in G$ there is an isometric embedding $\gamma\colon [0, d(x, y)] \to G$ such that $\gamma(0) = x$ and $\gamma(d(x, y)) = y$). This justifies why for any $y, y' \in G$, there is always at least one *shortest path* or *geodesic path* γ in G joining y, y' whose length satisfies $|\gamma| = d(y, y')$. We define an undirected *simple loop* γ of G to be a simple path that intersects itself only at the endpoints. We denote by $\mathcal{L}(G)$ the finite set of all simple loops of the finite graph G. Since a *metric tree* is a metric graph with no loops, $\mathcal{L}(G) = \emptyset$ for a metric tree G. For a metric graph G, let $e(G)$ denote the length of the shortest edge in G. If for $A \subseteq G$ there exists a unique shortest geodesic in G between any points $a, a' \in A$, then we define the *(geodesic) convex hull* of A in G to be the union of geodesics between all a and a'. If $\operatorname{diam}(A) < \frac{e(G)}{2}$, then the convex hull of A is well-defined. Given a (connected and finite) metric graph G, let G_0 be the smallest connected subgraph of G containing the union of all simple loops of G. We call G_0 the *core of* G. If G has no leaf edges, then $G = G_0$. Note that G_0 is just G with its "hanging trees" removed, or in other words, the minimal subspace onto which G deformation retracts. We will use the following two properties about G_0. First, if G (and hence G_0) contains at least two simple loops, then at least one vertex of G_0 has degree at least 3. We can then consider the canonical graph representation of G_0 consisting of only vertices of degree at least 3. Second, we note that if a path connected subset $Z \subseteq G$ has first Betti number equal to the first Betti number of G, then $G_0 \subseteq Z$.

[1] One needs to be careful if E is infinite. Suppose one has two vertices v and v', and one edge e_n of length $\frac{1}{n}$ with endpoints v and v' for each $n \in \mathbb{N}$. The quotient semi-metric gives $d(v, v') = \inf_{n \in \mathbb{N}} \frac{1}{n} = 0$ though $v \neq v'$; this semi-metric is not a metric. These subtleties disappear since we assume E is finite.

2.3 Hausdorff distances

Let (Z, d) be a metric space. If X and Y are two subsets of Z, then the *directed Hausdorff distance* from X to Y, denoted $\vec{d}_{\mathrm{H}}(X, Y)$, is defined by

$$\vec{d}_{\mathrm{H}}(X, Y) = \sup_{x \in X} \inf_{y \in Y} d(x, y) = \inf \left\{ r \geq 0 \mid X \subseteq Y^r \right\}.$$

Note that $\vec{d}_{\mathrm{H}}(X, Y)$ is not symmetric in general. To retain symmetry, the *Hausdorff distance*, denoted $d_{\mathrm{H}}(X, Y)$, is defined as $d_{\mathrm{H}}(X, Y) = \max \left\{ \vec{d}_{\mathrm{H}}(X, Y), \vec{d}_{\mathrm{H}}(Y, X) \right\}$. In other words, the Hausdorff distance finds the infimum over all real numbers r such that if we thicken Y by r it contains X, and if we thicken X by r it contains Y. If no such r exists then the Hausdorff distance is infinity.

2.4 Gromov–Hausdorff distances

Let (X, d_X) and (Y, d_Y) be metric spaces. In equations (1)–(3) we give three equivalent definitions of the Gromov–Hausdorff distance between X and Y. The first definition follows [14]. The *Gromov–Hausdorff distance* $d_{\mathrm{GH}}(X, Y)$ is the infimum of the Hausdorff distances between $F(X)$ and $G(Y)$ in Z, with the infimum being over all metric spaces Z and all isometric embeddings $F\colon X \hookrightarrow Z$ and $G\colon Y \hookrightarrow Z$:

$$d_{\mathrm{GH}}(X, Y) = \inf_{F, G, Z} d_{\mathrm{H}}(F(X), G(Y)). \tag{1}$$

The second definition uses correspondences. A relation $\mathcal{R} \subseteq X \times Y$ is a *correspondence* if for every $x \in X$ there is some $y \in Y$ with $(x, y) \in \mathcal{R}$, and for every $y \in Y$ there is some $x \in X$ exists with $(x, y) \in \mathcal{R}$. The *distortion* of a correspondence $\mathcal{R}$ is

$$\operatorname{dis}(\mathcal{R}) = \sup_{(x,y),(x',y') \in \mathcal{R}} \left| d_X(x, x') - d_Y(y, y') \right|.$$

The *Gromov–Hausdorff* distance between two (arbitrary) metric spaces X and Y is

$$d_{\mathrm{GH}}(X, Y) = \tfrac{1}{2} \inf_{\mathcal{R} \subseteq X \times Y} \operatorname{dis}(\mathcal{R}), \tag{2}$$

where the infimum is taken over all correspondences $\mathcal{R}$ between X and Y [9, 17]. For metric spaces (X, d_X) and (Y, d_Y), let $f\colon X \to Y$ and $g\colon Y \to X$ be two (possibly discontinuous) functions between them. We define the *distortion of* f as

$$\operatorname{dis}(f) = \sup_{x, x' \in X} \left| d_X(x, x') - d_Y(f(x), f(x')) \right|.$$

The distortion of g is defined similarly. The *codistortion of* f *and* g is defined as

$$\operatorname{codis}(f, g) = \sup_{x \in X, y \in Y} \left| d_X(x, g(y)) - d_Y(f(x), y) \right|.$$

Kalton and Ostrovskii [17] show that

$$d_{\mathrm{GH}}(X, Y) = \tfrac{1}{2} \inf_{f, g} \max\{\operatorname{dis}(f), \operatorname{dis}(g), \operatorname{codis}(f, g)\}, \tag{3}$$

where $f\colon X \to Y$ and $g\colon Y \to X$ are any functions. It follows that if $d_{\mathrm{GH}}(X, Y) < C$, then there is some function $f\colon X \to Y$ with $\frac{1}{2}\operatorname{dis}(f) < C$.

2.5 Gromov–Hausdorff distances and compactness

The following result is folklore; see [3] for the proof.

▶ **Lemma 2.** *Let X be a metric space and Y be a subset thereof. Let us denote by $\overline{Y}$ the closure of Y in X. Then*

$$d_{\mathrm{H}}(X,Y) = d_{\mathrm{H}}(X,\overline{Y}) \quad \text{and} \quad d_{\mathrm{GH}}(X,Y) = d_{\mathrm{GH}}(X,\overline{Y}).$$

Furthermore, for every two subsets Y and Z of X, we have $\vec{d}_{\mathrm{H}}(Z,Y) = \vec{d}_{\mathrm{H}}(Z,\overline{Y})$.

We will consistently apply Lemma 2 when the ambient metric space is compact. Thus, when we are studying the Hausdorff and the Gromov–Hausdorff distances between a compact metric space and a subspace thereof, we can assume that the subspace is compact as well.

2.6 Gromov–Hausdorff distances and connectedness

▶ **Lemma 3.** *Let $f\colon X \to Y$ be a function between metric spaces, with X path-connected and Y geodesic. Then for every $r > \frac{\operatorname{dis}(f)}{2}$, the thickening $f(X)^r$ is path connected.*

Proof. Let $y_1, y_2 \in f(X)^r$ and $x_1, x_2 \in X$ be points such that $d_Y(y_i, f(x_i)) < r$ for $i = 1, 2$. Since Y is geodesic, the entire geodesic connecting y_i to $f(x_i)$ is contained in $B(f(x_i), r) \subseteq f(X)^r$. Pick $\varepsilon > 0$ sufficiently small to satisfy $r > \frac{\operatorname{dis}(f)}{2} + \varepsilon$. Since X is path connected, there are $z_1 = x_1, z_2, \ldots, z_{n+1} = x_2$ such that $d(z_j, z_{j+1}) < \varepsilon$ for every $j = 0, \ldots, n$. Thus, $d(f(z_j), f(z_{j+1})) < \operatorname{dis}(f) + \varepsilon$. Since Y is geodesic, there are $m_j \in Y$ such that $d(f(z_j), m_j) = d(f(z_{j+1}), m_j) = \frac{1}{2} d(f(z_j), f(z_{j+1})) < r$ and the geodesics connecting m_j to $f(z_j)$ and $f(z_{j+1})$ are contained in $f(X)^r$. ◀

2.7 Gromov–Hausdorff distances and graphs

Let G be a finite metric graph and let $X \subseteq G$ be a compact subset. To use functions $f\colon G \to X$ to prove bounds on the Gromov–Hausdorff distance, we need to be a bit careful. Indeed, even if the sample X is close to the leaves of G, it may be that $f(G)$ is not close to the leaves. Luckily, we have a lot of freedom to construct our function f:

▶ **Lemma 4.** *Let G be a finite metric graph and let $X \subseteq G$ be compact. If $d_{\mathrm{GH}}(G, X) < r$, then there is a function $f\colon G \to X$ with $\frac{1}{2}\operatorname{dis}(f) < r$ and $\vec{d}_{\mathrm{H}}(\partial G, f(G)) = \vec{d}_{\mathrm{H}}(\partial G, X)$.*

Proof. Let $\mathcal{R} \subseteq G \times X$ be a correspondence satisfying $\operatorname{dis}(\mathcal{R}) < 2r$. Fix $\varepsilon > 0$ such that $\operatorname{dis}(\mathcal{R}) + \varepsilon < 2r$. For every $v \in \partial G$, pick $x_v \in X$ as a point satisfying $d(v, x_v) = \min_{x \in X} d(v, x)$. We first recursively construct a surjective map $\widetilde{f}$ from a subset of G to $\{x_v \mid v \in \partial G\}$. Let us arbitrarily order the leaves $v_1, \ldots, v_n$ and denote the point x_{v_i} by x_i for the sake of simplicity. **Base step.** Pick any $y_1 \in G$ such that $(y_1, x_1) \in \mathcal{R}$ and set $\widetilde{f}(y_1) = x_1$. **Recursive step.** Assume we have defined $y_1, \ldots, y_m \in G$ distinct and set $\widetilde{f}(y_i) = x_i$. Pick $y \in G$ satisfying $(y, x_{m+1}) \in \mathcal{R}$. Let us distinguish two cases. If $y \notin \{y_1, \ldots, y_m\}$, then set $y_{m+1} = y$ and $\widetilde{f}(y_{m+1}) = x_{m+1}$. Otherwise, fix an arbitrary point $y_{m+1} \in G \setminus \{y_1, \ldots, y_m\}$ satisfying $d(y_{m+1}, y) < \frac{\varepsilon}{2}$ and define $\widetilde{f}(y_{m+1}) = x_{m+1}$. Consider an arbitrary extension $f\colon G \to X$ of $\widetilde{f}\colon \{y_1, \ldots, y_n\} \to \{x_1, \ldots, x_n\}$ with the property that the graph of $f|_{G \setminus \{y_1, \ldots, y_n\}}$ is contained in $\mathcal{R}$. Then $\operatorname{dis}(f) \le \operatorname{dis}(\mathcal{R}) + 2 \cdot \frac{\varepsilon}{2} < 2r$ and $\vec{d}_{\mathrm{H}}(\partial G, f(G)) = \vec{d}_{\mathrm{H}}(\partial G, X)$ by construction. ◀

It is worth noticing that in the proof above we did not use any property of ∂G, and the result can be shown with ∂G replaced by any finite subset Y of G.

3 Ratio of d_{GH} vs d_{H} can be arbitrarily small for metric trees

In this section we prove Theorem 5, a variant of Theorem 5 of [2], now adapted to metric trees. As we show in Theorems 8 and 23, the Gromov–Hausdorff distance $d_{\mathrm{GH}}(T, X)$ between a metric graph T and a subset $X \subseteq T$ can be bounded below by their Hausdorff distance $d_{\mathrm{H}}(T, X)$ when X is close enough to the leaves of T, i.e., when $d_{\mathrm{H}}(T, X) > \vec{d}_{\mathrm{H}}(\partial T, X)$. When $d_{\mathrm{H}}(T, X) < \vec{d}_{\mathrm{H}}(\partial T, X)$, however, the ratio of $d_{\mathrm{GH}}(T, X)$ over $d_{\mathrm{H}}(T, X)$ can become arbitrarily small as we show below. Furthermore, Theorem 5 suggests that ∂T may not map close to itself via correspondences with small distortion; see Figure 1.

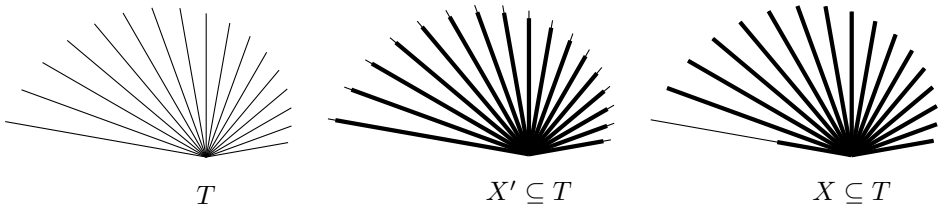


Figure 1 A sketch of the construction of Theorem 5: (left) The metric tree T, (middle) the bold superimposed $X' \subseteq T$ with small $d_{\mathrm{H}}(T, X')$, and (right) a different isometric embedding of X' into T indicated by $X \subseteq T$ with large $d_{\mathrm{H}}(T, X)$.

▶ **Theorem 5.** *For any $\varepsilon > 0$ there exists a finite metric tree T and a closed subset $X \subseteq T$ such that $d_{\mathrm{GH}}(T, X) \leq \varepsilon \cdot d_{\mathrm{H}}(T, X)$.*

Proof. Choose $n \in \mathbb{N}$ and let $\varepsilon = \frac{1}{n}$. For $t \geq 0$, let $I(t)$ be a copy of the interval $[0, t]$ with a designated basepoint 0. Define a metric tree T as the wedge of n intervals of lengths $1 + \varepsilon, 1 + 2\varepsilon, \ldots, 1 + n\varepsilon = 2$, and define a metric tree X_0 as the wedge of intervals of lengths $1, 1 + \varepsilon, 1 + 2\varepsilon, \ldots, 1 + (n-1)\varepsilon = 2 - \varepsilon$:

$$T = \bigvee_{t=1}^{n} I(1 + t \cdot \varepsilon) \quad \text{and} \quad X_0 = \bigvee_{t=0}^{n-1} I(1 + t \cdot \varepsilon).$$

Recall from (1) that $d_{\mathrm{GH}}(T, X)$ is the infimum of the Hausdorff distances between $F(T)$ and $G(X)$ in Z, with the infimum being over all metric spaces Z and all isometric embeddings $F \colon T \hookrightarrow Z$ and $G \colon X \hookrightarrow Z$. In Figure 1 we have two different isometric embeddings X', X of the same tree X_0 into $Z = T$: one with a small Hausdorff distance $d_{\mathrm{H}}(T, X') = \varepsilon$ as X' is T with each ray being shortened by ε, and one with a large Hausdorff distance $d_{\mathrm{H}}(T, X) = 1$ as X is T with the longest ray $I(2)$ of length 2 being replaced with a subray $I(1)$ of length 1. By (1), we have $d_{\mathrm{GH}}(T, X) \leq d_{\mathrm{H}}(T, X') = \varepsilon d_{\mathrm{H}}(T, X)$. ◀

4 Metric trees with coefficient 1

Section 3 demonstrates that results claiming $d_{\mathrm{GH}}(G, X) = d_{\mathrm{H}}(G, X)$ for X a subset of a metric graph G do not hold in general, even if $d_{\mathrm{H}}(G, X)$ is arbitrarily small. In this section, as well as in Section 6, we prove related results relying on bounds on $\vec{d}_{\mathrm{H}}(\partial G, X)$. In particular, we will assume the condition $d_{\mathrm{H}}(G, X) > \vec{d}_{\mathrm{H}}(\partial G, X)$ which rules out the example constructed in Theorem 5. The following proposition shows that condition has a high probability of being satisfied if X is a large uniform sample of G; see [3] for the proof.

▶ **Proposition 6.** *If G is a metric graph and X is a sample of n uniformly random points, then with high probability as $n \to \infty$, we have $d_{\mathrm{H}}(G, X) > \vec{d}_{\mathrm{H}}(\partial G, X)$.*

We prove the following results for metric trees.

▶ **Proposition 7.** *Let T be a finite tree, and let $f\colon T \to T$ be a function. If $d_{\mathrm{H}}(T, f(T)) > \vec{d}_{\mathrm{H}}(\partial T, f(T))$, then $\frac{1}{2}\mathrm{dis}(f) \geq d_{\mathrm{H}}(T, f(T))$.*

Proof. Let $\frac{1}{2}\mathrm{dis}(f) < r$ and let $\overline{x} \in T$ be a point such that $B(\overline{x}, d_{\mathrm{H}}(T, f(T))) \cap f(T) = \emptyset$. The assumption $d_{\mathrm{H}}(T, f(T)) > \vec{d}_{\mathrm{H}}(\partial T, f(T))$ implies that $T \setminus \{\overline{x}\}$ is disconnected and at least two distinct components thereof have non-empty intersection with $f(T)$ (since a tree has at least two leaves). Since $f(T)^r$ is connected by Lemma 3, we have $\overline{x} \in f(T)^r$, which implies $d_{\mathrm{H}}(T, f(T)) < r$. Since r can be taken arbitrarily, $d_{\mathrm{H}}(T, f(T)) \leq \frac{1}{2}\mathrm{dis}(f)$. ◀

▶ **Theorem 8.** *Let T be a finite metric tree and let $X \subseteq T$. If $d_{\mathrm{H}}(T, X) > \vec{d}_{\mathrm{H}}(\partial T, X)$, then $d_{\mathrm{GH}}(T, X) = d_{\mathrm{H}}(T, X)$.*

Proof. By Lemma 2, we may assume that X is compact. Suppose for a contradiction that $d_{\mathrm{GH}}(T, X) < d_{\mathrm{H}}(T, X)$. Then by Lemma 4, there is a function $f\colon T \to X \subseteq T$ with $\frac{1}{2}\mathrm{dis}(f) < d_{\mathrm{H}}(T, X) \leq d_{\mathrm{H}}(T, f(T))$ and $\vec{d}_{\mathrm{H}}(\partial T, f(T)) = \vec{d}_{\mathrm{H}}(\partial T, X) < d_{\mathrm{H}}(T, X) \leq d_{\mathrm{H}}(T, f(T))$, contradicting Proposition 7. ◀

In other words, $d_{\mathrm{H}}(T, X) = \max\{d_{\mathrm{GH}}(T, X), \vec{d}_{\mathrm{H}}(\partial T, X)\}$. All of the results in our paper have simple corollaries for two subsets of the tree or graph; we give one such example.

▶ **Corollary 9.** *Let T be a finite metric tree and let $X, Y \subseteq T$. For any $\varepsilon \geq d_{\mathrm{H}}(T, Y)$, if $d_{\mathrm{H}}(X, Y) > \vec{d}_{\mathrm{H}}(\partial T, X) + \varepsilon$, then $d_{\mathrm{GH}}(X, Y) \geq d_{\mathrm{H}}(X, Y) - 2\varepsilon$.*

Proof. We need only apply the triangle inequality:

$$
\begin{aligned}
& d_{\mathrm{H}}(X, Y) \leq d_{\mathrm{H}}(T, X) + d_{\mathrm{H}}(T, Y) \\
&= \max\left\{d_{\mathrm{GH}}(T, X), \vec{d}_{\mathrm{H}}(\partial T, X)\right\} + d_{\mathrm{H}}(T, Y) && \text{by Theorem 8} \\
&\leq \max\left\{d_{\mathrm{GH}}(X, Y) + d_{\mathrm{GH}}(T, Y), \vec{d}_{\mathrm{H}}(\partial T, X)\right\} + d_{\mathrm{H}}(T, Y) \\
&\leq \max\left\{d_{\mathrm{GH}}(X, Y) + d_{\mathrm{H}}(T, Y), \vec{d}_{\mathrm{H}}(\partial T, X)\right\} + d_{\mathrm{H}}(T, Y) && \text{since } d_{\mathrm{GH}}(T, Y) \leq d_{\mathrm{H}}(T, Y) \\
&= \max\left\{d_{\mathrm{GH}}(X, Y) + 2d_{\mathrm{H}}(T, Y), \vec{d}_{\mathrm{H}}(\partial T, X) + d_{\mathrm{H}}(T, Y)\right\} \\
&\leq \max\left\{d_{\mathrm{GH}}(X, Y) + 2\varepsilon, \vec{d}_{\mathrm{H}}(\partial T, X) + \varepsilon\right\} && \text{since } d_{\mathrm{H}}(T, Y) \leq \varepsilon.
\end{aligned}
$$
◀

As a particular case of Theorem 8, we can provide a closed formula for the Gromov–Hausdorff distance between a segment and a sample of it; see [3] for the proof.

▶ **Corollary 10.** *Let X be a compact non-empty subset of an interval $I = [a, b]$. Denote by $c = \min X$ and $d = \max X$. Then $d_{\mathrm{GH}}(I, X) = \max\left\{\frac{c-a+b-d}{2}, d_{\mathrm{H}}([c, d], X)\right\}$.*

5 The case of the circle

Before considering general metric graphs in the following section, in this section we first understand the simplest connected metric graph which is not a tree: the circle $G = S^1$. The authors of [2] showed that $d_{\mathrm{GH}}(S^1, X) = d_{\mathrm{H}}(S^1, X)$ for a compact subset $X \subseteq S^1$ with $d_{\mathrm{GH}}(S^1, X) < \frac{\pi}{6}$. A curious question regarding the optimality of the density constant $\frac{\pi}{6}$ was also posed therein [2, Question 2]. While the constant $\frac{\pi}{6}$ provides a sufficient Gromov–Hausdorff density of a subset X to guarantee the equality of its Hausdorff and Gromov–Hausdorff distances to S^1, it turns out this constant is suboptimal. In this section,

we show that $\frac{\pi}{3}$ is the optimal constant in the case of the circle. That is, for a subset $X \subseteq S^1$, the density condition $d_{\mathrm{GH}}(S^1, X) < \frac{\pi}{3}$ implies $d_{\mathrm{GH}}(S^1, X) = d_{\mathrm{H}}(S^1, X)$ (Theorem 11), and there is a compact $X \subseteq S^1$ with $\frac{\pi}{3} = d_{\mathrm{GH}}(S^1, X) < d_{\mathrm{H}}(S^1, X)$ (Theorem 12).

▶ **Theorem 11.** *For any subset $X \subseteq S^1$, we have $d_{\mathrm{GH}}(S^1, X) \geq \min\left\{d_{\mathrm{H}}(S^1, X),\ \frac{\pi}{3}\right\}$.*

Proof. This proof is closely related to [18, 12]. It suffices to show that if $d_{\mathrm{GH}}(S^1, X) < \frac{\pi}{3}$, then $d_{\mathrm{GH}}(S^1, X) \geq d_{\mathrm{H}}(S^1, X)$. Let $f\colon S^1 \to X$ be a (possibly discontinuous) function. By (3), it suffices to show that if $\frac{1}{2}\mathrm{dis}(f) < \frac{\pi}{3}$, then $\frac{1}{2}\mathrm{dis}(f) \geq d_{\mathrm{H}}(S^1, X)$, or equivalently that $\frac{\varepsilon + \mathrm{dis}(f)}{2} \geq d_{\mathrm{H}}(S^1, X)$ for all $\varepsilon > 0$ sufficiently small. Suppose $\frac{1}{2}\mathrm{dis}(f) < \frac{\pi}{3}$. Fix $\varepsilon > 0$ with $5\varepsilon < 2\pi - 3\mathrm{dis}(f)$. Form a finite triangulation K of S^1 with edges of length less than ε. Let $K^{(0)} \subseteq S^1$ be the vertex set of K. Since adjacent vertices $v_i, v_{i+1} \in K^{(0)}$ satisfy $d(v_i, v_{i+1}) < \varepsilon$, we have $d(f(v_i), f(v_{i+1})) < \varepsilon + \mathrm{dis}(f) < \pi$. Hence we can map the edge between v_i and v_{i+1} to the unique shortest path in S^1 between $f(v_i)$ and $f(v_{i+1})$, obtaining a continuous map $\overline{f}\colon K \to S^1$. Let $r = \frac{\varepsilon + \mathrm{dis}(f)}{2}$. Suppose for a contradiction that X^r is

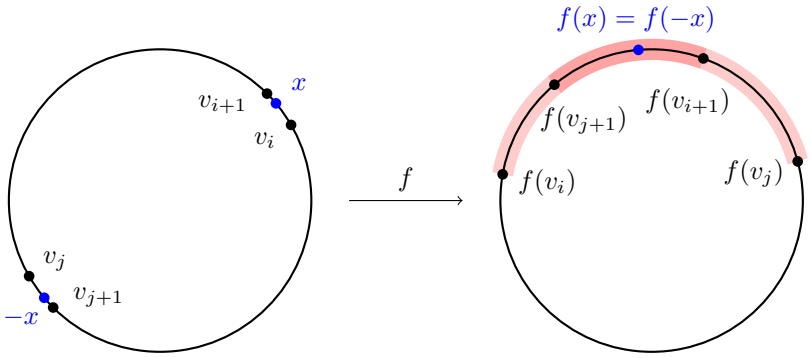


Figure 2 Proof of Theorem 11: Since $\overline{f}(x) = \overline{f}(-x)$, the shortest paths in S^1 between $f(v_i), f(v_{i+1})$ and $f(v_j), f(v_{j+1})$ intersect.

not equal to all of S^1. Then there is an arc of length at least $2r$ in S^1 that contains no points from X. So $\overline{f}$ is non-surjective, since the image of each edge in K is an edge of length less than $2r$. Hence $\overline{f}\colon K \to S^1$ is a map of degree zero. The Borsuk–Ulam theorem [5, Page 206, Exercise 21] then implies there are antipodal points $x, -x \in K$ with $\overline{f}(x) = \overline{f}(-x)$. Let x be on the closed edge between $v_i, v_{i+1} \in K^{(0)}$, and let $-x$ be on the closed edge between $v_j, v_{j+1} \in K^{(0)}$. Since $\overline{f}(x) = \overline{f}(-x)$, the shortest path in S^1 between $f(v_i), f(v_{i+1})$ intersects the shortest path between $f(v_j), f(v_{j+1})$. Since these paths in S^1 are of length less than $2r$, one of $f(v_i), f(v_{i+1})$ is at a distance less than r from one of $f(v_j), f(v_{j+1})$, i.e. $d(f(v), f(v')) < r$ for some $v \in \{v_i, v_{i+1}\}$ and $v' \in \{v_j, v_{j+1}\}$; see Figure 2. We obtain

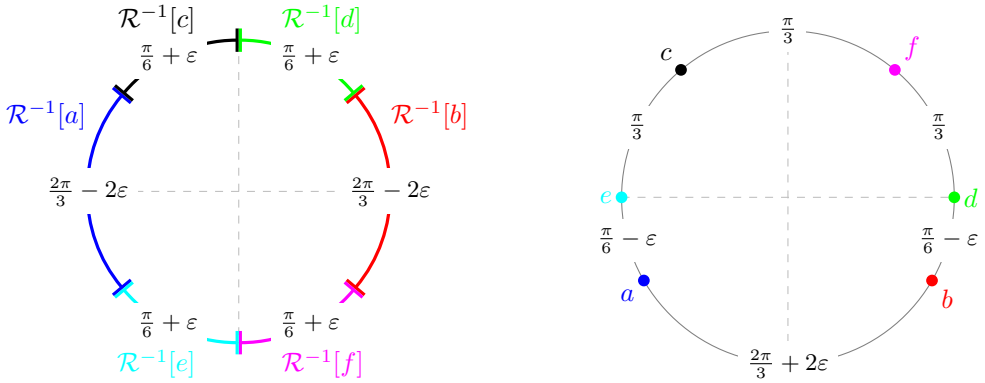
$$\mathrm{dis}(f) > d(v, v') - d(f(v), f(v')) > (\pi - 2\varepsilon) - \tfrac{\varepsilon + \mathrm{dis}(f)}{2} = \pi - \tfrac{\mathrm{dis}(f)}{2} - \tfrac{5\varepsilon}{2}.$$

So $\frac{5\varepsilon}{2} > \pi - \frac{3\mathrm{dis}(f)}{2}$, contradicting the choice of ε. Hence it must be that $X^r = S^1$. Since $\frac{\varepsilon + \mathrm{dis}(f)}{2} = r \geq d_{\mathrm{H}}(S^1, X)$ for all $\varepsilon > 0$ sufficiently small, we are done. ◀

We now prove the optimality of $\frac{\pi}{3}$ for the circle, i.e. that $\frac{\pi}{3}$ is also a necessary condition.

▶ **Theorem 12.** *For any $\varepsilon \in \left(0, \frac{\pi}{6}\right)$, there exists a nonempty compact subset $X \subseteq S^1$ with $d_{\mathrm{H}}(S^1, X) = \frac{\pi}{3} + \varepsilon$ and $d_{\mathrm{GH}}(S^1, X) = \frac{\pi}{3} < d_{\mathrm{H}}(S^1, X)$.*

Proof. We construct X to be a six-element subset of S^1. Both X and a particular correspondence $\mathcal{R} \subseteq S^1 \times X$ are shown in Figure 3. One can check that $d_{\mathrm{H}}(S^1, X) = \frac{\pi}{3} + \varepsilon$. Also, one can check that $\mathrm{dis}(\mathcal{R}) = \frac{2\pi}{3}$, giving $d_{\mathrm{GH}}(S^1, X) \leq \frac{\pi}{3}$. So $d_{\mathrm{GH}}(S^1, X) < d_{\mathrm{H}}(S^1, X)$. Moreover, Theorem 11 implies that $d_{\mathrm{GH}}(S^1, X) = \frac{\pi}{3}$. ◀



(a) Correspondence $\mathcal{R} \subseteq S^1 \times X$ has a distortion of $\frac{2\pi}{3}$. So, $d_{\mathrm{GH}}(S^1, X) \leq \frac{\pi}{3}$.

(b) $X = \{a, b, c, d, e, f\} \subseteq S^1$ with $d_{\mathrm{H}}(S^1, X) = \frac{\pi}{3} + \varepsilon$.

Figure 3 The configuration of points $X \subseteq S^1$ is depicted to show the optimal constant for the circle.

6 Metric graphs with loops with coefficient 1

In this section, we extend our results for the circle to general metric graphs; see Theorem 23.

6.1 Distortion of continuous maps between graphs

We first need a Borsuk–Ulam theorem for maps into trees; see [3] for the proof.

▶ **Lemma 13.** *If T is a tree and $f\colon S^1 \to T$ is continuous, then there exist $x, x' \in S^1$ with $d(x, x') \geq \frac{2\pi}{3}$ and $f(x) = f(x')$.*

Note that the lower bound $\frac{2\pi}{3}$ in Lemma 13 cannot be improved, as shown in Figure 4. Recall

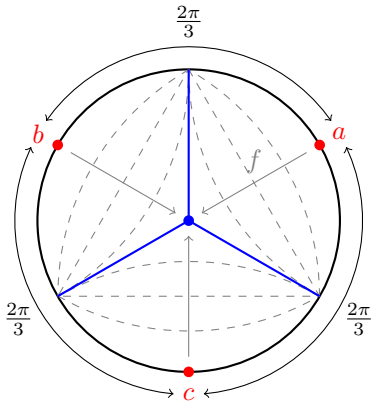


Figure 4 The lower bound in Lemma 13 is tight. The function f sends the black circle to the blue tripod. The points a, b and c are at distance $\frac{2\pi}{3}$ apart and in the same fiber of f.

that G_0 is the core of a metric graph G (see Section 2), and that $e(G_0)$ is the length of the shortest edge in G_0.

▶ **Lemma 14.** *Let (G, d) be a metric graph and γ be a simple loop. If $y, y' \in \gamma$ satisfy $d(y, y') < \frac{e(G_0)}{2}$, then there is a unique geodesic joining y and y' and it lies on γ.*

Proof. Let η be a simple path connecting y and y'. If η is not entirely in γ, its length is at least $e(G_0)$. So since $d(y, y') < \frac{e(G_0)}{2}$, a geodesic connecting the two points must lie on γ; see Figure 5. This geodesic is unique since the length of a loop is at least $e(G_0)$. ◀

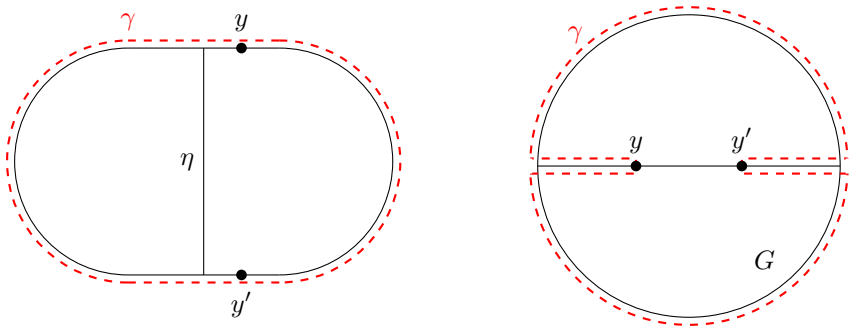


Figure 5 The assumptions in Lemma 14 are necessary. Loop γ is represented with a dashed red line. (a) If there is no bound on $e(G_0)$, then there may exist a shorter path connecting y and y' using a shortcut η. (b) If γ is not simple, then y and y' may be arbitrarily close and joined by geodesics that are not in γ.

The following lemma is the crucial step showing that a continuous map between finite graphs must send simple loops to simple loops, provided that the distortion is sufficiently small compared to the length of the shortest internal edge.

▶ **Lemma 15.** *Let $\overline{f}\colon (G, d_G) \to (G', d_{G'})$ be a continuous map between finite metric graphs. Assume that $\frac{\mathrm{dis}(\overline{f})}{2} < \frac{\min\{e(G_0), e(G_0')\}}{8}$. Then $\overline{f}$ induces an injection $\Phi\colon \mathcal{L}(G) \to \mathcal{L}(G')$ between the simple loops of G and the simple loops of G' as follows: given a simple loop $\gamma \in \mathcal{L}(G)$, $\Phi(\gamma)$ is the only simple loop of G' contained in $\overline{f}(\gamma)$.*

Proof of Lemma 15 □. For readability purposes, let us divide the proof into several claims. First, let us show that the map Φ is well-defined, i.e., for every simple loop $\gamma \in \mathcal{L}(G)$ there is a simple loop of G' contained in $\overline{f}(\gamma)$ and that simple loop is unique.

▷ Claim 16 (Existence). If γ is a simple loop of G, then $\overline{f}(\gamma)$ contains a simple loop of G'.

Proof of Claim 16. Suppose for a contradiction that $\overline{f}(\gamma)$ does not contain a loop. Since $\overline{f}(\gamma)$ is connected, it is a tree. So Lemma 13 applies to give points $x, x' \in \gamma$ with $\overline{f}(x) = \overline{f}(x')$ and $d_\gamma(x, x') \geq \frac{\mathrm{length}(\gamma)}{3}$ where d_γ is the geodesic distance on γ. We must have $d_G(x, x') \geq \frac{e(G_0)}{3}$, since if $d_G(x, x') < \frac{e(G_0)}{3} < \frac{e(G_0)}{2}$, then Lemma 14 would give that a geodesic connecting x and x' lies on γ, and so $d_G(x, x') = d_\gamma(x, x') \geq \frac{\mathrm{length}(\gamma)}{3} \geq \frac{e(G_0)}{3}$. We obtain the contradiction

$$\mathrm{dis}(\overline{f}) \geq |d_G(x, x') - d_{G'}(\overline{f}(x), \overline{f}(x'))| = d_G(x, x') \geq \frac{e(G_0)}{3} > \frac{e(G_0)}{4} > \mathrm{dis}(\overline{f}).$$

Hence $\overline{f}(\gamma)$ must contain a loop. ◁

▷ Claim 17 (Uniqueness). If γ is a simple loop, then $\overline{f}(\gamma)$ contains at most one simple loop.

Proof of Claim 17. Note that $\overline{f}(\gamma)$ is a closed subset of G' since it is the continuous image of a compact subset. Let $H = \overline{f}(\gamma)$, which is a connected subgraph of G'. Furthermore, let H_0 be the smallest connected subgraph of G' containing the union of all simple loops of H. Suppose for a contradiction that $\overline{f}(\gamma) = H$ contains at least two simple loops. Therefore,

at least one vertex of H_0 has degree at least 3. We can then consider the canonical graph representation of H_0 consisting of only vertices of degree at least 3. For every $v \in V(H_0)$, the closed subset $(\overline{f})^{-1}(v)$ has diameter at most $\mathrm{dis}(\overline{f}) < \frac{e(G_0)}{2}$. Hence, the convex hull C_v of $(\overline{f})^{-1}(v) \cap \gamma$ is well-defined. Furthermore, for every $v, v' \in V(H_0)$ with $v \neq v'$, we have $C_v \cap C_{v'} = \emptyset$. Indeed, for $x \in (\overline{f})^{-1}(v)$ and $x' \in (\overline{f})^{-1}(v')$ with $v \neq v'$, we have

$$d_G(x, x') \geq d_{G'}(v, v') - \mathrm{dis}(\overline{f}) \geq e(G'_0) - \mathrm{dis}(\overline{f}) > \mathrm{dis}(\overline{f}) \geq \max\{\mathrm{diam} C_v, \mathrm{diam} C_{v'}\}.$$

Let $|V(H_0)| = n$ and let $\gamma_1, \ldots, \gamma_n$ be the non-empty connected components of $\gamma \setminus \bigcup_{v \in V(H_0)} C_v$. For every $i \in \{1, \ldots, n\}$, note that $\overline{f}(\gamma_i)$ is contained in a connected component of $H \setminus V(H_0)$. Let $\eta_1, \ldots, \eta_m$ be the connected components of $H_0 \setminus V(H_0)$. Note that each η_j is an open-

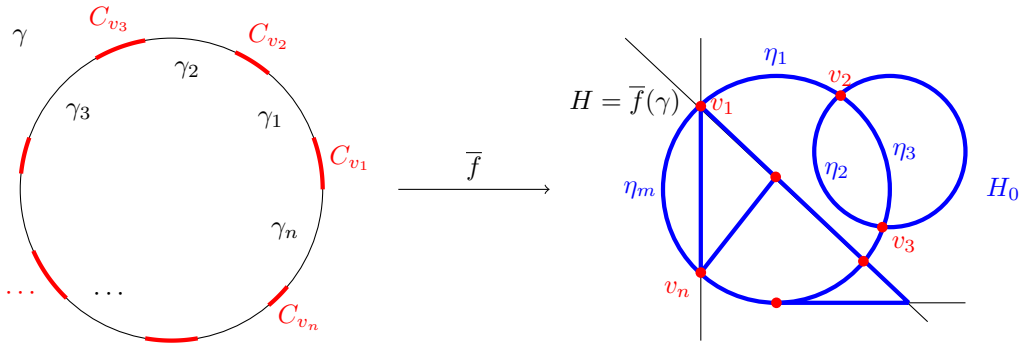


Figure 6 A representation of the objects constructed in the proof of Claim 17.

ended path in H_0. We claim that $\eta_j \cap \overline{f}(\gamma_1 \cup \cdots \cup \gamma_n) \neq \emptyset$ for every $j \in \{1, \ldots, m\}$. Let y be the midpoint of η_j. Since $H = \overline{f}(\gamma)$, there is $z \in \gamma$ such that $\overline{f}(z) = y$. For every $v \in V(H_0)$ and every $x \in (\overline{f})^{-1}(v)$,

$$d_G(z, x) \geq d_{G'}(y, v) - \mathrm{dis}(\overline{f}) \geq \tfrac{e(G'_0)}{2} - \mathrm{dis}(\overline{f}) > 2\mathrm{dis}(\overline{f}) - \mathrm{dis}(\overline{f}) = \mathrm{dis}(\overline{f}) \geq \mathrm{diam} C_v.$$

Hence, $z \notin C_v$ and so $z \in \gamma_1 \cup \cdots \cup \gamma_n$. So far, we have shown that each $\overline{f}(\gamma_i)$, for $i \in \{1, \ldots, n\}$, is contained in some connected component of $H \setminus V(H_0)$. Furthermore, among these connected components, each η_j with $j \in \{1, \ldots, m\}$ contains the $\overline{f}$-image of at least one γ_i. It will be a contradiction if we prove that $n < m$. Note that the set $\{\eta_j\}_{j=1}^m$ is in bijection with the edge set $E(H_0)$ of H_0. Hence

$$2m = 2|E(G'_0)| = \sum_{v \in V(G'_0)} \deg_{G'_0}(v) \geq 3|V(G'_0)| = 3n,$$

giving the contradiction $n < m$. It must be that $\overline{f}(\gamma)$ contains at most one simple loop. ◁

According to Claims 16 and 17, map Φ is well-defined. Lastly, we show that it is injective.

▷ Claim 18. The map $\Phi\colon \mathcal{L}(G) \to \mathcal{L}(G')$ is injective.

Proof of Claim 18. Let γ_1, γ_2, and η be three simple loops such that $\eta \subseteq \overline{f}(\gamma_1) \cap \overline{f}(\gamma_2)$. Suppose, by contradiction, that the simple loops γ_1 and γ_2 are distinct, meaning they differ by at least one edge. Without loss of generality, let us fix an edge $e \in E(G)$ that is contained in γ_1 but whose interior is disjoint from γ_2. Fix the midpoint m of e. In particular,

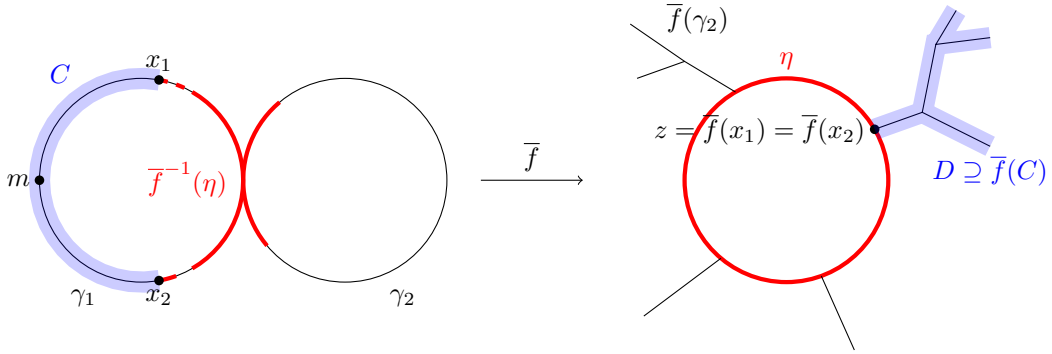


Figure 7 A representation of the objects constructed in the proof of Claim 18.

$d_G(m, \gamma_2) \geq \frac{e(G_0)}{2}$. Let us estimate $d_G(m, (\overline{f})^{-1}(\eta))$. Suppose that $x \in (\overline{f})^{-1}(\eta)$, and fix $y \in \gamma_2$ such that $\overline{f}(x) = \overline{f}(y)$. Note that $d_G(x, y) \leq \mathrm{dis}(\overline{f})$. Using the triangle inequality, we obtain

$$\begin{aligned} d_G(m, x) &\geq d_G(m, y) - d_G(x, y) \geq d_G(m, \gamma_2) - \mathrm{dis}(\overline{f}) \\ &\geq \tfrac{e(G_0)}{2} - \mathrm{dis}(\overline{f}) > 2\mathrm{dis}(\overline{f}) - \mathrm{dis}(\overline{f}) = \mathrm{dis}(\overline{f}). \end{aligned} \tag{4}$$

Hence $d(m, (\overline{f})^{-1}(\eta)) \geq \mathrm{dis}(\overline{f})$, and in particular, $\overline{f}(m) \notin \eta$. Consider the connected component C of $\gamma_1 \setminus (\overline{f})^{-1}(\eta)$ containing m. Since $(\overline{f})^{-1}(\eta)$ is closed by continuity, C is an open arc in γ_1. Let x_1 and x_2 be the two points of γ_1 on the boundary of C. Note that they both belong to $(\overline{f})^{-1}(\eta)$, and so $d_G(m, x_i) > \mathrm{dis}(\overline{f})$ by (4). Let us estimate $d_G(x_1, x_2)$. A geodesic connecting x_1 and x_2 satisfies one of the following:

(i) either it passes through m,
(ii) or it crosses $(\overline{f})^{-1}(\eta) \cap \gamma_1$,
(iii) or it uses an edge that is not in γ_1.

In all three cases, we have $d_G(x_1, x_2) > \mathrm{dis}(\overline{f})$. For (i), this follows since $d_G(m, x_i) > \mathrm{dis}(\overline{f})$. For (ii), let $A \coloneqq (\overline{f})^{-1}(\eta) \cap \gamma_1$. Since $\overline{f}(\gamma_1)$ contains η, we have $\overline{f}(A) = \eta$. The diameter of η is at least $\frac{e(G_0')}{2}$, and hence $\mathrm{diam}(A) \geq \frac{e(G_0')}{2} - \mathrm{dis}(\overline{f}) > 2\mathrm{dis}(\overline{f}) - \mathrm{dis}(\overline{f}) = \mathrm{dis}(\overline{f})$.

Finally, we also have $d_G(x_1, x_2) > \mathrm{dis}(\overline{f})$ in case (iii) since $e(G_0) > \mathrm{dis}(\overline{f})$. Since $\overline{f}$ is continuous, $\overline{f}(C)$ is contained in a connected component D of $\overline{f}(\gamma_1) \setminus \eta$. Since $\overline{f}(\gamma_1)$ contains only one loop, D is a tree. Furthermore, $\overline{f}(\gamma_1)$ is a closed subset of G' since it is the continuous image of a compact subset, and it contains only one simple loop. Thus, the closure of D in G' intersects η in only one point z. We have already observed that $\overline{f}(x_i) \in \eta$ for $i \in \{1, 2\}$. Furthermore, $\overline{f}(x_i)$ belongs to the closure of $\overline{f}(C)$ in G', which is contained in the closure of D. Therefore, $\overline{f}(x_i) = z$ since the latter is the only point in the closure of D that lies on η. This gives $\overline{f}(x_1) = z = \overline{f}(x_2)$, which contradicts the above established inequality $d_G(x_1, x_2) > \mathrm{dis}(\overline{f})$. ◁

This completes the proof of Lemma 15. ◀

6.2 Distortion of arbitrary functions between graphs

To apply Lemma 15 in the context of Gromov–Hausdorff distances, we will need a way to turn arbitrary functions into nearby continuous ones with controlled distortion, as described in the following result.

▶ **Lemma 19.** *Let $f\colon (G, d_G) \to (G', d_{G'})$ be a function between finite metric graphs with distortion $\frac{\mathrm{dis}(f)}{2} < r < \frac{\min\{e(G_0), e(G_0')\}}{4}$. Then there is a continuous map $\overline{f}\colon G \to G'$ such that $\frac{\mathrm{dis}(\overline{f})}{2} < 2r$ and $\overline{f}(G) \subseteq (f(G))^r$.*

Proof. We organize the proof into three steps, to help its readability. Let $\varepsilon > 0$ satisfy $\varepsilon + \operatorname{dis}(f) < 2r$.

Step 1: Construct a continuous map $\overline{f}$. Fix a finite triangulation K of G with edges of length at most ε such that $V(G) \subseteq K^{(0)}$. Let us construct a continuous map $\overline{f}\colon K \to G'$ as follows. For every vertex $x \in K^{(0)}$, set $\overline{f}(x) = f(x)$. Consider now two adjacent vertices x and y in $K^{(0)}$. Send the segment from x to y to the geodesic segment connecting $f(x)$ and $f(y)$ at uniform speed. The geodesic between the two points $f(x)$ and $f(y)$ is unique since $d_{G'}(f(x), f(y)) \leq \varepsilon + \operatorname{dis}(f) < \frac{e(G'_0)}{2}$.

Step 2: Estimate the distortion of $\overline{f}$. We claim that $\operatorname{dis}(\overline{f}) \leq 2(\operatorname{dis}(f) + \varepsilon) < 4r$. The second inequality is trivial. Let $x, y \in K$. Choose a closest point $x_i \in K^{(0)}$ to x. Let $x_{i+1} \in K^{(0)}$ be such that x is contained in the edge $\{x_i, x_{i+1}\}$. Note that $d_G(x, x_i) \leq \frac{\varepsilon}{2}$. Thus, by construction of $\overline{f}$,

$$d_{G'}(\overline{f}(x), f(x_i)) \leq \frac{d_{G'}(f(x_i), f(x_{i+1}))}{2} \leq \frac{\varepsilon + \operatorname{dis}(f)}{2} \tag{5}$$

Similarly, if $y_j \in K^{(0)}$ is a closest point to y, $d_G(y, y_j) \leq \frac{\varepsilon}{2}$ and $d_{G'}(\overline{f}(y), f(y_j)) \leq \frac{\varepsilon + \operatorname{dis}(f)}{2}$. By applying the triangle inequality, we obtain the following chain of inequalities:

$$\begin{aligned}
&d_{G'}(\overline{f}(x), \overline{f}(y)) \leq d_{G'}(\overline{f}(x), \overline{f}(x_i)) + d_{G'}(f(x_i), f(y_j)) + d_{G'}(\overline{f}(y_j), \overline{f}(y)) \\
&\leq \tfrac{\operatorname{dis}(f)+\varepsilon}{2} + d_G(x_i, y_j) + \operatorname{dis}(f) + \tfrac{\operatorname{dis}(f)+\varepsilon}{2} \leq 2\operatorname{dis}(f) + \varepsilon + d_G(x_i, x) + d_G(x, y) + d_G(y, y_j) \\
&\leq 2(\operatorname{dis}(f) + \varepsilon) + d_G(x, y) < 4r + d_G(x, y).
\end{aligned}$$

With similar computations, we can show that $d_G(x, y) - d_{G'}(\overline{f}(x), \overline{f}(y)) \leq 2(\operatorname{dis}(f) + \varepsilon) < 4r$. This proves $\frac{\operatorname{dis}(\overline{f})}{2} < 2r$.

Step 3: Show $\overline{f}(K) \subseteq (f(G))^r$. For every point $\overline{f}(x) \in \overline{f}(K)$, choose a closest point $x_i \in K^{(0)}$ to x as in **Step 2**. According to (5), $d_{G'}(\overline{f}(x), f(x_i)) < r$ since $\varepsilon + \operatorname{dis}(f) < 2r$. ◀

We finally obtain that the core G_0 of a graph G is preserved by a function with distortion bounded by a fraction of the shortest edge length.

▶ **Proposition 20.** *Let G be a finite metric graph and let $f\colon G \to G$ be a function with $\frac{\operatorname{dis}(f)}{2} < r < \frac{e(G_0)}{16}$. Then $G_0 \subseteq f(G_0)^r$.*

Proof. Let $\overline{f}\colon G \to G$ be the continuous map described in Lemma 19. Then $\frac{\operatorname{dis}(\overline{f})}{2} < 2r < \frac{e(G_0)}{8}$. The restriction $\overline{f}|_{G_0}\colon G_0 \to G$ is also continuous and has the same bound on its distortion. We can then apply Lemma 15 to $\overline{f}|_{G_0}$ and obtain an injection $\Phi\colon \mathcal{L}(G_0) \to \mathcal{L}(G)$. Since $\mathcal{L}(G_0) = \mathcal{L}(G)$ by construction and these sets are finite, Φ is a bijection. Hence, every simple loop of G is contained in $\overline{f}(G_0) \subseteq f(G_0)^r$ according to Lemma 19. Given that $f(G_0)^r$ is connected, it contains G_0. ◀

As an immediate consequence of Proposition 20 we obtain that the Hausdorff and the Gromov–Hausdorff distances between a graph with no leaves and a subset thereof coincide, provided that the latter is sufficiently small.

▶ **Corollary 21.** *Let G be a finite metric graph with no leaves, and let $X \subseteq G$. If $d_{\mathrm{GH}}(G, X) < \frac{e(G_0)}{16}$, then $d_{\mathrm{GH}}(G, X) = d_{\mathrm{H}}(G, X)$.*

Proof. Let $r > 0$ satisfy $d_{\mathrm{GH}}(G, X) < r < \frac{e(G_0)}{16}$. Then there is a map $f\colon G \to X$ whose distortion satisfies $\frac{\mathrm{dis}(f)}{2} < r < \frac{e(G_0)}{16}$. We can apply Proposition 20 to obtain that $G = G_0 \subseteq f(G_0)^r \subseteq X^r$. Hence $d_{\mathrm{H}}(G, X) \leq r$. Since r can be taken arbitrarily, we have $d_{\mathrm{H}}(G, X) \leq d_{\mathrm{GH}}(G, X)$. ◀

We now discuss how to lower bound the Gromov–Hausdorff distance with respect to the Hausdorff distance for graphs with leaves. The first result relies on hypotheses around the subset X of the graph G. More precisely, similarly to the tree case, we request that the subset X is sufficiently close to the leaves of G.

▶ **Proposition 22.** *Let (G, d) be a finite metric graph and let $f\colon G \to G$ be a (possibly discontinuous) function with $\frac{\mathrm{dis}(f)}{2} < \frac{e(G_0)}{16}$. If $\partial G \neq \emptyset$, suppose that $d_{\mathrm{H}}(G, f(G)) > \vec{d}_{\mathrm{H}}(\partial G, f(G))$. Then $d_{\mathrm{H}}(G, f(G)) \leq \frac{\mathrm{dis}(f)}{2}$.*

Proof. Let $r > 0$ satisfy $\frac{\mathrm{dis}(f)}{2} < r < \frac{e(G_0)}{16}$. Let $\overline{x} \in G$ be a point such that the intersection $B(\overline{x}, d_{\mathrm{H}}(G, f(G))) \cap f(G)$ is empty. If $\overline{x} \in G_0$, then $\overline{x} \in f(G)^r$ according to Proposition 20, and so $d_{\mathrm{H}}(G, f(G)) < r$. Suppose otherwise that $\overline{x} \notin G_0$. Thus, $G \setminus \{\overline{x}\}$ is disconnected, and at least two connected components thereof have non-empty intersection with X by the assumption. We can conclude as in Proposition 7. ◀

▶ **Theorem 23.** *Let G be a finite connected metric graph and let $X \subseteq G$. If $\partial G \neq \emptyset$, suppose $d_{\mathrm{H}}(G, X) > \vec{d}_{\mathrm{H}}(\partial G, X)$. If $d_{\mathrm{GH}}(G, X) < \frac{e(G_0)}{16}$, then $d_{\mathrm{GH}}(G, X) = d_{\mathrm{H}}(G, X)$.*

Proof. As in the proof of Theorem 8, by Lemma 2 we can assume that X is compact by replacing it with its closure $\overline{X}$ if necessary. Suppose, by contradiction, that $d_{\mathrm{GH}}(G, X) < d_{\mathrm{H}}(G, X)$. By Lemma 4 there is a function $f\colon G \to X$ such that $\frac{1}{2}\mathrm{dis}(f) < \min\{d_{\mathrm{H}}(G, X), \frac{e(G_0)}{16}\}$ and $\vec{d}_{\mathrm{H}}(\partial G, X) = \vec{d}_{\mathrm{H}}(\partial G, f(G))$. Note that $d_{\mathrm{H}}(G, f(G)) \geq d_{\mathrm{H}}(G, X) > \vec{d}_{\mathrm{H}}(\partial G, X) = \vec{d}_{\mathrm{H}}(\partial G, f(G))$. We can apply Proposition 22 to obtain a contradiction with the following chain of inequalities: $d_{\mathrm{H}}(G, X) \leq d_{\mathrm{H}}(G, f(G)) \leq \frac{\mathrm{dis}(f)}{2} \leq d_{\mathrm{GH}}(G, X)$. ◀

7 Conclusion and open questions

We have furthered the study of lower bounding the Gromov–Hausdorff distance between a space and a subset by their Hausdorff distance, particularly for metric graphs. This investigation sparks several open questions and new research directions, especially for spaces beyond graphs: (1) Is the density constant $\frac{e(G_0)}{16}$ in Theorem 23 optimal? (2) Do our results generalize to general length spaces? If so, under what density assumptions? (3) Are there versions of our results that hold for manifolds with boundary? (4) Are there classes of metric spaces (other than graphs) where the Gromov–Hausdorff distance between the space and a dense enough subset equals their Hausdorff distance?

References

1 Henry Adams, Johnathan Bush, Nate Clause, Florian Frick, Mario Gómez, Michael Harrison, R. Amzi Jeffs, Evgeniya Lagoda, Sunhyuk Lim, Facundo Mémoli, Michael Moy, Nikola Sadovek, Matt Superdock, Daniel Vargas, Qingsong Wang, and Ling Zhou. Gromov–Hausdorff distances, Borsuk–Ulam theorems, and Vietoris–Rips complexes. *arXiv preprint*, 2023. `arXiv:2301.00246`.

2 Henry Adams, Florian Frick, Sushovan Majhi, and Nicholas McBride. Hausdorff vs Gromov–Hausdorff distances. *Discrete & Computational Geometry*, 2025.

3 Henry Adams, Sushovan Majhi, Fedor Manin, Žiga Virk, and Nicolò Zava. Lower bounding the Gromov–Hausdorff distance in metric graphs. arXiv:2411.09182.

4 Pankaj K. Agarwal, Kyle Fox, Abhinandan Nath, Anastasios Sidiropoulos, and Yusu Wang. Computing the Gromov–Hausdorff distance for metric trees. *ACM Trans. Algorithms*, 14(2), 2018. doi:10.1145/3185466.

5 Mark Anthony Armstrong. *Basic topology*. Springer Science & Business Media, 2013.

6 Martin R Bridson and André Haefliger. *Metric spaces of non-positive curvature*, volume 319. Springer Science & Business Media, 2011.

7 L. E. J. Brouwer. Beweis der Invarianz der Dimensionenzahl. *Mathematische Annalen*, 70(2):161–165, 1911.

8 L. E. J. Brouwer. Über Abbildung von Mannigfaltigkeiten. *Mathematische Annalen*, 71(1):97–115, 1911.

9 Dmitri Burago, Yuri Burago, and Sergei Ivanov. *A course in metric geometry*, volume 33. American Mathematical Society, Providence, 2001.

10 Gunnar Carlsson and Facundo Mémoli. Characterization, stability and convergence of hierarchical clustering methods. *Journal of Machine Learning Research*, 11(47):1425–1470, 2010. doi:10.5555/1756006.1859898.

11 Frédéric Chazal, Vin de Silva, and Steve Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 174:193–214, 2014.

12 Lester Dubins and Gideon Schwarz. Equidiscontinuity of Borsuk–Ulam functions. *Pacific Journal of Mathematics*, 95(1):51–59, 1981.

13 David A Edwards. The structure of superspace. In *Studies in topology*, pages 121–133. Elsevier, 1975.

14 Mikhael Gromov. Structures métriques pour les variétés Riemanniennes. *Textes Mathématiques*, 1, 1981.

15 Michael Harrison and R Amzi Jeffs. Quantitative upper bounds on the Gromov–Hausdorff distance between spheres. *arXiv preprint*, 2023. arXiv:2309.11237.

16 Yibo Ji and Alexey A Tuzhilin. Gromov–Hausdorff distance between segment and circle. *arXiv preprint*, 2021. arXiv:2101.05762.

17 Nigel J Kalton and Mikhail I Ostrovskii. Distances between Banach spaces. *Forum Math.*, 11:1–17, 1999.

18 Alejandro Leon. The Gromov–Hausdorff distance between spheres and their boundaries. To appear, 2026.

19 Sunhyuk Lim, Facundo Mémoli, and Zane Smith. The Gromov–Hausdorff distance between spheres. *Geometry & Topology*, 27:3733–3800, 2023.

20 Sushovan Majhi, Jeffrey Vitter, and Carola Wenk. Approximating Gromov–Hausdorff distance in Euclidean space. *Computational Geometry*, 116:102034, 2024. doi:10.1016/j.comgeo.2023.102034.

21 Facundo Mémoli. On the use of Gromov–Hausdorff distances for shape comparison, 2007.

22 Facundo Mémoli. Some properties of Gromov–Hausdorff distances. *Discrete & Computational Geometry*, 48(2):416–440, 2012. doi:10.1007/S00454-012-9406-8.

23 Felix Schmiedl. Computational aspects of the Gromov–Hausdorff distance and its application in non-rigid shape matching. *Discrete & Computational Geometry*, 57(4):854, 2017.

24 Alexey A Tuzhilin. Who invented the Gromov–Hausdorff distance? *arXiv preprint*, 2016. arXiv:1612.00728.

25 Nicolò Zava. Coarse and bi-Lipschitz embeddability of subspaces of the Gromov–Hausdorff space into Hilbert spaces. *Algebr. Geom. Topol.*, 25(8):5153–5174, 2024.

Dynamic Nearest-Neighbor Searching Under General Metrics in $\mathbb{R}^3$ and Its Applications

Pankaj K. Agarwal
Department of Computer Science, Duke University, Durham, NC, USA

Matthew J. Katz
The Stein Faculty of Computer and Information Science,
Ben-Gurion University of the Negev, Beer Sheva, Israel

Micha Sharir
School of Computer Science, Tel Aviv University, Tel Aviv, Israel

Abstract

Let K be a compact, centrally-symmetric, strictly-convex region in $\mathbb{R}^3$, which is a semi-algebraic set of constant complexity, i.e. the unit ball of a corresponding metric, denoted as $\|\cdot\|_K$. Let $\mathcal{K}$ be a set of n homothetic copies of K. This paper contains two main sets of results:

(i) For a storage parameter $s \in [n, n^3]$, $\mathcal{K}$ can be preprocessed in $O^*(s)$ expected time into a data structure of size $O^*(s)$, so that for a query homothet K_0 of K, an intersection-detection query (determine whether K_0 intersects any member of $\mathcal{K}$, and if so, report such a member) or a nearest-neighbor query (return the member of $\mathcal{K}$ whose $\|\cdot\|_K$-distance from K_0 is smallest) can be answered in $O^*(n/s^{1/3})$ time; all k homothets of $\mathcal{K}$ intersecting K_0 can be reported in additional $O(k)$ time. In addition, the data structure supports insertions/deletions in $O^*(s/n)$ amortized expected time per operation. Here the $O^*(\cdot)$ notation hides factors of the form n^ε, where $\varepsilon > 0$ is an arbitrarily small constant, and the constant of proportionality depends on ε.

(ii) Let $\mathcal{G}(\mathcal{K})$ denote the intersection graph of $\mathcal{K}$. Using the above data structure, breadth-first or depth-first search on $\mathcal{G}(\mathcal{K})$ can be performed in $O^*(n^{3/2})$ expected time. Combining this result with the so-called shrink-and-bifurcate technique, the reverse-shortest-path problem in a suitably defined proximity graph of $\mathcal{K}$ can be solved in $O^*(n^{62/39})$ expected time. Dijkstra's shortest-path algorithm, as well as Prim's MST algorithm, on a $\|\cdot\|_K$-proximity graph on n points in $\mathbb{R}^3$, with edges weighted by $\|\cdot\|_K$, can also be performed in $O^*(n^{3/2})$ time.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Homothets, Minkowski metric, Shallow cuttings, Nearest-neighbor searching, Intersection and proximity graphs, Reverse-shortest-path problem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.4

Related Version *Full Version*: `https://arxiv.org/abs/2603.26585`

Funding *Pankaj K. Agarwal*: Partially supported by NSF grants CCF-2223870 and IIS-2402823, and by a US-Israel Binational Science Foundation Grant 2022/131.
Matthew J. Katz: Partially supported by Israel Science Foundation Grant 495/23.
Micha Sharir: Partially supported by Israel Science Foundation Grant 495/23.

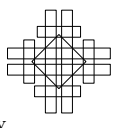
1 Introduction

Problem statement. Let K be a compact, centrally-symmetric, strictly-convex region in $\mathbb{R}^3$, which is a semi-algebraic set of constant complexity.[1] K is the unit ball of a norm $\|\cdot\|_K$, and we denote its metric as dist_K.

[1] Roughly speaking, a semi-algebraic set in $\mathbb{R}^d$ is the set of points in $\mathbb{R}^d$ that satisfy a Boolean formula over a set of polynomial inequalities; the complexity of a semi-algebraic set is the number of polynomials defining the set and their maximum degree. See [12] for formal definitions of a semi-algebraic set and its complexity.



© Pankaj K. Agarwal, Matthew J. Katz, and Micha Sharir;
licensed under Creative Commons License CC-BY 4.0





A *homothet* (or *homothetic copy*) of K is a scaled and translated copy of K, expressed as $K(c,\rho) := \rho K + c$, for $c \in \mathbb{R}^3$ and $\rho \geq 0$; c is the *center* of $K(c,\rho)$ and ρ is its *size*. We represent $K(c,\rho)$ as the point $(c,\rho) \in \mathbb{R}^3 \times \mathbb{R}_{\geq 0}$. Let $\mathbb{K}$ be the set of all homothets of K, represented as the halfspace $x_4 \geq 0$ of $\mathbb{R}^4$. The *K-distance* $\mathrm{dist}_K(u,v) = \|u-v\|_K$ between points $u, v \in \mathbb{R}^3$ is the minimum value of ρ for which $v \in K(u,\rho)$ (or, equivalently, $u \in K(v,\rho)$). Since K is strictly convex, the triangle inequality $\mathrm{dist}_K(u,v) \leq \mathrm{dist}_K(u,w) + \mathrm{dist}_K(w,v)$ is sharp unless w lies on the segment uv. The K-distance of a point $x \in \mathbb{R}^3$ from a homothet $K(c,\rho)$ is defined as $f_{c,\rho}(x) = \mathrm{dist}_K(c,x) - \rho$, and the K distance of another homothet $K(c',\rho')$ from $K(c,\rho)$ is $f_{c,\rho}(c') - \rho' = \mathrm{dist}_K(c,c') - \rho - \rho'$. Since K is a constant-complexity semi-algebraic set, $f_{c,\rho}$ is a semi-algebraic function of constant complexity. The homothets $K(c,\rho)$ and $K(c',\rho')$ intersect if the distance between them is non-positive, i.e., $\mathrm{dist}_K(c,c') \leq \rho + \rho'$, or equivalently, the point (c',ρ') lies above the graph of $f_{c,\rho}$ in $\mathbb{R}^4$, or vice-versa.

Let $\mathcal{K} = \{K_1, \ldots, K_n\} \subset \mathbb{K}$ be a set of n homothets of K, where $K_i = K(c_i, \rho_i)$. In this paper we study two classes of proximity problems related to $\mathcal{K}$.

Proximity queries. We wish to preprocess $\mathcal{K}$ into a data structure so that various proximity queries for a query homothet K_0, such as intersection-detection (determine whether K_0 intersects any homothet of $\mathcal{K}$; if the answer is yes, return one such intersecting homothet of $\mathcal{K}$), intersection reporting (report all homothets of $\mathcal{K}$ that intersect K_0), and nearest-neighbor queries (report the homothet of $\mathcal{K}$ at the minimum K-distance from K_0), can be performed efficiently. In addition, the data structure should support insertions and deletions of homothets of K.

Searching in the intersection graph and related problems. Let $\mathcal{G}(\mathcal{K})$ denote the intersection graph of $\mathcal{K}$, whose vertices are the homothets of $\mathcal{K}$ and (K_i, K_j) is an edge if $K_i \cap K_j \neq \emptyset$. Similarly, for a threshold parameter r_0, the *r_0-proximity graph* $\mathcal{G}_{r_0}(\mathcal{K})$ of $\mathcal{K}$ consists of those edges (K_i, K_j) for which $\mathrm{dist}_K(K_i, K_j) \leq r_0$. We study efficient implementation of graph-search problems on $\mathcal{G}(\mathcal{K})$, like BFS or DFS. We also study the *reverse shortest path* (RSP) problem on $\mathcal{K}$: given two elements $K_i, K_j \in \mathcal{K}$ and a parameter $1 \leq k < n$, compute the smallest value r^* for which $\mathcal{G}_{r^*}(\mathcal{K})$ contains a path from K_i to K_j with at most k edges.

Related work. There is extensive work on nearest-neighbor (NN) searching in many different fields including computational geometry, database systems, and machine learning. We refer the readers to various survey papers [4, 10, 20] for a general overview of known results on this topic. Here we briefly mention a few results that are most closely related to the problems studied in this paper. Agarwal and Matoušek [9] presented a dynamic NN data structure for a point set in $\mathbb{R}^d$, under the Euclidean metric, that answers a query in $O^*(n/s^\phi)$ time, where $\phi = 1 - \frac{1}{\lceil d/2 \rceil}$ and $s \in [n, n^{\lceil d/2 \rceil}]$ is a storage parameter, using $O^*(s)$ space and preprocessing; the (amortized) update time for insertions/deletions of points is $O^*(s/n)$. The bounds were slightly improved – n^ε factors were replaced by $\log^{O(1)} n$ factors – in [15,26]. Kaplan *et al.* [26] presented a dynamic NN data structure for a set S of points in $\mathbb{R}^2$ under fairly general distance functions. Its performance depends on the complexity of the Voronoi diagram of S under that distance function. For the Minkowski metric induced by a centrally-symmetric convex region, which is a semi-algebraic set of constant complexity, their data structure can answer an NN query in $O(\log^2 n)$ time, and handle insertion and deletion of a point in $O(\beta(n)\log^5 n)$ and $O(\beta(n)\log^9 n)$ amortized expected time, respectively, where $\beta(n)$, a variant of inverse Ackermann's function, is an extremely slowly growing function; see also [31] for slightly improved bounds. These data structures, however, do not extend to $\mathbb{R}^3$.

The problem of NN searching under general distance functions in $\mathbb{R}^3$ is much less understood. By reducing NN queries to ball-intersection queries [8] and using semi-algebraic range-searching data structures [1, 2, 33], an NN query amid the set $\mathcal{K}$ of n homothets in $\mathbb{R}^3$, as above, can be answered in $O^*(1)$ time using $O^*(n^4)$ space and preprocessing, or in $O^*(n^{3/4})$ time using $O(n)$ space and preprocessing. Using recent standard techniques (see, e.g., the Appendix in [1]), one can obtain a space/query-time trade-off, as well as handle insertions/deletions of objects. Using a recent result by Agarwal *et al.* [5] on the vertical decomposition of the lower envelope of trivariate functions, an NN query amid $\mathcal{K}$ can be answered in $O(\log^2 n)$ time using only $O^*(n^3)$ space, but this data structure does not support efficient deletions (insertions can be handled in a standard manner, using the dynamization technique of Bentley and Saxe [13]). Furthermore, it was not known whether an NN query amid $\mathcal{K}$ can be answered in $O^*(n^{2/3})$ time using $O^*(n)$ space.

Motivated by numerous applications, there has been work on developing fast algorithms for intersection and proximity graphs of n geometric objects. Although the intersection graph may have $\Theta(n^2)$ edges, the goal in this line of work is to develop algorithms, by exploiting the underlying geometry, that run in $O^*(n)$ time (or at least in subquadratic time). Cabello and Jejčič [14], and subsequently Chan and Skrepetos [21], presented $O(n \log n)$ implementations of BFS in unit-disk intersection graphs, which was recently extended to general disk-intersection graphs by de Berg and Cabello [22]; see also [25, 26, 29]. The algorithm in [22] can also perform DFS and Dijkstra's algorithm in $O(n \log n)$ time. Katz *et al.* [28] showed that BFS in ball-intersection graphs (for congruent or arbitrary Euclidean balls) can be implemented in $O^*(n^{2\phi/(\phi+1)})$ time, where $\phi = \lfloor d/2 \rfloor + 1$, so in $O^*(n^{4/3})$ time for $d = 3$. The problem of devising efficient implementations for BFS/DFS in more general graphs in the plane also has received some attention, see, e.g., [6, 7, 27].

The reverse shortest path (RSP) problem for unit disks was studied by Wang and Zhao [36], who gave an $O^*(n^{5/4})$-time solution. Kaplan *et al.* [25] improved the runtime to $O^*(n^{6/5})$ using the *shrink-and-bifurcate* technique (see [11]), and it was recently improved to $O^*(n^{9/8})$ by Chan and Huang [17]. The best-known RSP algorithm for arbitrary disks runs in $O^*(n^{6/5})$ expected time. Katz *et al.* [28] presented RSP algorithms for balls in $\mathbb{R}^3$ – with $O^*(n^{29/21})$ time for congruent balls and $O^*(n^{56/39})$ for arbitrary balls. The RSP problem also has been studied for other geometric objects in $\mathbb{R}^2$ [7, 17, 28].

Our results. Our main result is a dynamic data structure for answering intersection and NN queries with a homothet of $\mathbb{K}$ amid the set $\mathcal{K}$ of homothets:

▶ **Theorem 1.** *Let K be a compact, centrally-symmetric strictly convex semi-algebraic set in $\mathbb{R}^3$ of constant complexity. Let $\mathcal{K}$ be a set of n homothetic copies of K. For a storage parameter $s \in [n, n^3]$, $\mathcal{K}$ can be maintained in a dynamic data structure with $O^*(s)$ storage, that can answer an intersection-detection or NN-query (under the K-distance) with a homothet K_0 of K in $O^*(n/s^{1/3})$ time and that supports insertions/deletions in $O^*(s/n)$ amortized expected time. It can report all k objects of $\mathcal{K}$ intersecting K_0 in additional $O(k)$ time.*

Plugging Theorem 1 into Eppstein's technique [23], or its enhancement by Chan [16], for maintaining bichromatic closest pairs, using $s = n^{3/2}$, we obtain:

▶ **Corollary 2.** *Let K be a compact, centrally-symmetric strictly-convex semi-algebraic set in $\mathbb{R}^3$ of constant complexity. Let $\mathcal{K}, \mathcal{K}'$ be two sets of homothets of K of combined size n. The K-closest pair between $\mathcal{K}$ and $\mathcal{K}'$, under insertions/deletions of homothets (in $\mathcal{K}$ and $\mathcal{K}'$), can be maintained in $O^*(n^{1/2})$ amortized expected time per update.*

Theorem 1 is obtained by presenting two data structures. The first one (see Section 4.1) uses $O^*(n^3)$ space, answers a query in $O^*(1)$ time, and handles an insertion/deletion in $O^*(n^2)$ amortized expected time. (The data structure in [5] also obtains a similar space/query-time bound but it cannot handle deletions efficiently.) This data structure is based on constructing *vertical shallow cuttings*, a notion originally introduced in [15,18], of the graphs of the distance functions of $\mathcal{K}$. Roughly speaking, for a parameter $t \geq 1$, a *vertical* (n/t)*-shallow cutting* of $\mathcal{K}$ is a collection of pairwise openly disjoint semi-unbounded *pseudo-prisms* that covers the region lying below the k-level of the arrangement of $\mathcal{K}$, where each prism, consisting of all points that lie vertically below a constant-complexity semi-algebraic 3-dimensional region, intersects the graphs of only $O(n/t)$ functions of $\mathcal{K}$. The random-sampling based technique used in [15, 26] to construct a vertical shallow cutting does not immediately extend to our setting because one needs to decompose the q-level, for some parameter $q > 0$, in the 4D arrangement of the distance functions of $\mathcal{K}$ into $O^*(n^3(q+1))$ constant-complexity cells. Although a recent result [5] constructs such a decomposition of the 0-level (with $O^*(n^3)$ cells), it does not extend to larger values of q. This problem remains elusive for arrangements of arbitrary trivariate semi-algebraic functions of constant complexity, but we develop a desired decomposition technique for our setting (in Section 2) by exploiting the structural geometric properties of the distance functions of $\mathcal{K}$. This result is one of the main technical contributions of the paper. Using our result on the decomposition of the q-level, we construct a vertical (n/t)-shallow cutting of $\mathcal{K}$ of size $O^*(t^3)$, for any t (Section 3).

The second data structure is a linear-size partition tree, with $O^*(n^{2/3})$ query time, constructed on the point set $\mathcal{K}^* = \{(c_i, \rho_i) \mid 1 \leq i \leq n\} \subset \mathbb{R}^4$, that answers intersection-detection queries on $\mathcal{K}$ with a homothet of $\mathbb{K}$ as a query. The main technical challenge we face here is the construction of a so-called *test set* of $\mathcal{K}$ (see [35] and below) of size $r^{O(1)}$, which represents well the distance functions of all (n/r)*-shallow* homothets in $\mathbb{K}$, i.e., the homothets that intersect at most n/r elements of $\mathcal{K}$. By adapting the technique in [35] and again exploiting the structural geometric properties of the distance functions, we show that a test set, of size $O^*(r^4)$, with the desired properties can be constructed efficiently.

Using Theorem 1 and Corollary 2, we obtain our second set of results:

▶ **Theorem 3.**

(a) *Given a set $\mathcal{K}$ of n homothets of a constant-complexity, strictly convex, centrally-symmetric semi-algebraic set K, BFS or DFS in the intersection graph of $\mathcal{K}$ can be performed in $O^*(n^{3/2})$ expected time.*

(b) *The single-source shortest-path tree as well as the minimum spanning forest of the r_0-proximity graph of a set of n points, for any threshold parameter r_0, weighted by the pairwise K-distances, can be computed in $O^*(n^{3/2})$ time.*

Combining Theorem 3(a) with the shrink-and-bifurcate technique [11, 25] yields:

▶ **Theorem 4.** *Given a set $\mathcal{K}$ of n homothets of a constant-complexity, strictly convex, centrally-symmetric semi-algebraic set K, two designated elements $K, K' \in \mathcal{K}$, and an integer $k < n$, the RSP problem on $\mathcal{K}$, of finding the smallest r^* for which $G_{r^*}(\mathcal{K})$ contains a path between K and K' of length at most k, can be solved in $O^*(n^{62/39})$ expected time.*

Because of lack of space, the proofs of Theorems 3 and 4 and of some of the lemmas are omitted from this version.

2 Decomposing the k-Level of an Arrangement of Distance Functions

Preliminaries. Let $\mathcal{K} = \{K_1, \ldots, K_n\}$, where each K_i is a homothetic copy of K represented by a point $K_i^* = (c_i, \rho_i)$ in $\mathbb{R}^4$. For $1 \le i \le n$, set $f_i = f_{c_i,\rho_i}$, and let $\mathcal{F} := \mathcal{F}(\mathcal{K}) = \{f_i \mid 1 \le i \le n\}$. (As noted, distances from points x that lie inside some copy K_i, still measured by $f_i(x)$, are nonpositive.) We will not distinguish between a function of $\mathcal{F}$ and its graph. For a point $x \in \mathbb{R}^3$, K_i is the *nearest neighbor* of x in $\mathcal{K}$ (under the K-distance dist_K), i.e., $i = \arg\min_{1\le j\le n} f_j(x)$, if f_i appears on the lower envelope of $\mathcal{F}$ at x.

The *level* of a point $p \in \mathbb{R}^4$ in $\mathcal{A}(\mathcal{F})$ is the number of functions in $\mathcal{F}$ whose graphs lie below p. For a parameter $k < n$, the *k-level* of $\mathcal{A}(\mathcal{F})$, denoted $\mathcal{A}_k(\mathcal{F})$, is the (closure of the) locus of all points on $\bigcup\mathcal{F}$ whose level is k. We define the *$(\le k)$-level* of $\mathcal{A}(\mathcal{F})$, denoted $\mathcal{A}_{\le k}(\mathcal{F})$, to be the (closure of the) set of all points in $\mathbb{R}^4$ of level at most k, i.e., the set of points that lie on or below $\mathcal{A}_k(\mathcal{F})$ (not necessarily on $\cup\mathcal{F}$). The projection of $\mathcal{A}_k(\mathcal{F})$ onto $\mathbb{R}^3$, denoted by $\mathcal{M}_k = \mathcal{M}_k(\mathcal{K})$, is a subdivision of $\mathbb{R}^3$. If a cell $\tau^\downarrow$ of $\mathcal{M}_k$ is the projection of a cell τ of $\mathcal{A}_k(\mathcal{F})$ that lies on the graph of f_i, then K_i is the k-th nearest neighbor in $\mathcal{K}$ (under the K-distance function) of all points in $\tau^\downarrow$. Note that $A_0(\mathcal{F})$ is the lower envelope of $\mathcal{F}$, and $\mathcal{M}_0(\mathcal{K})$ is the Voronoi diagram of $\mathcal{K}$ (under the K-distance).

The decomposition. The main technical result of this section is that $\mathcal{A}_k(\mathcal{F})$ can be decomposed into $O^*(n^3(k+1))$ *elementary cells*, namely, constant-complexity semi-algebraic regions, each homeomorphic to a ball; see below for a more precise definition.

For $1 \le i \ne j \le n$, let $B_{ij} = \{x \in \mathbb{R}^3 \mid f_i(x) = f_j(x)\}$ be the *bisector* of K_i and K_j under the K-distance. Note that if $K_i \subset K_j$ then, as is easily verified, $f_j(x) < f_i(x)$ for all $x \in \mathbb{R}^3$ and B_{ij} is undefined. The following lemma gives a useful property of bisectors.

▶ **Lemma 5.** *Fix a homothet $K_i \in \mathcal{K}$. Let λ be a ray in $\mathbb{R}^3$ emanating from c_i. For any $1 \le j \ne i \le n$ such that $K_i \not\subset K_j$, λ intersects B_{ij} in at most one point, say, ξ. Furthermore, $f_i(x) \le f_j(x)$ for all $x \in c_i\xi$ (where $c_i\xi$ denotes the segment with endpoints c_i and ξ) and $f_j(x) > f_i(x)$ for all $x \in \lambda \setminus c_i\xi$.*

We are now ready to describe the decomposition of the cells of $\mathcal{A}_k(\mathcal{F})$ into elementary cells. (Our ultimate goal is to decompose the region below $\mathcal{A}_k(\mathcal{F})$ into elementary cells, but we start with this subtask.) For $1 \le i \le n$, let $\mathcal{A}_k^i = \mathcal{A}_k^i(\mathcal{F})$ denote the subset of cells of $\mathcal{A}_k(\mathcal{F})$ that lie on the graph of f_i, and let $\mathcal{M}_k^i$ denote their projections (which are the cells of $\mathcal{M}_k$ for which K_i is the k-th nearest neighbor). We describe the algorithm for decomposing the cells of $\mathcal{M}_k^i$ (or of $\mathcal{A}_k^i$). Lemma 5 implies that, for any $j \ne i$, B_{ij} can be viewed as the graph of a function $g_{ij} : \mathbb{S}^2 \to \mathbb{R}_{\ge 0}$ in spherical coordinates with c_i as the origin. That is, for a given $u \in \mathbb{S}^2$, let $\lambda(u)$ be the ray emanating from c_i in direction u, and let $x_{ij}(u)$ be the intersection point of $\lambda(u)$ with the bisector B_{ij} if such an intersection point exists; otherwise $x_{ij}(u)$ is undefined. We set $g_{ij}(u) = f_j(x_{ij}(u))$ if $x_{ij}(u)$ is defined and $+\infty$ otherwise. Let $G^{(i)} = \{g_{ij} \mid 1 \le j \ne i \le n\}$. We define the *level* of a point $x \in \mathbb{R}^3$ in (the 3D arrangement, within $\mathbb{S}^2 \times \mathbb{R}$) $\mathcal{A}(G^{(i)})$ to be the number of functions in $G^{(i)}$ whose graphs intersect the (relatively open) segment c_ix. The following lemma will be useful in analyzing the complexity of the decomposition of $\mathcal{A}_k^i$.

▶ **Lemma 6.** *For a parameter $k < n$, let τ be a 3D cell of $\mathcal{A}_k^i$. Then $\tau^\downarrow$ is a cell of level at most k in $\mathcal{A}(G^{(i)})$.*

Proof. It is obvious from the definition of $G^{(i)}$ that $\tau^\downarrow$ is a cell in $\mathcal{A}(G^{(i)})$. We now argue that its level in $\mathcal{A}(G^{(i)})$ is at most k. Choose a point $x \in \tau^\downarrow$. Let λ be the ray emanating from c_i and passing through x. Let g_{ij} be a function whose graph intersects the segment

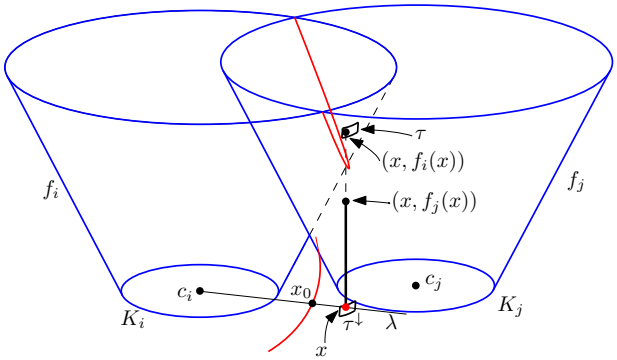


Figure 1 A bisector crossing λ. (The figure is drawn in the 3D c-space.)

$c_i x$, say, at a point $x_0 \in B_{ij}$. (See Fig. 1.) By Lemma 5, $f_i(y) < f_j(y)$ for all points y on λ preceding x_0, and $f_j(y) < f_i(y)$ for all points on λ beyond x_0. In particular, $f_i(x) > f_j(x)$. Since τ is a cell of $\mathcal{A}(\mathcal{F})$, f_j does not intersect τ, so f_j lies below τ in $\mathbb{R}^4$ (in the x_4-, or rather ρ-direction). Furthermore, $\tau \in \mathcal{A}_k(\mathcal{F})$, so there are exactly k functions of $\mathcal{F}$ whose graphs lie below τ, implying that the level of $\tau^\downarrow$ in $\mathcal{A}(G^{(i)})$ is at most k, as claimed. ◀

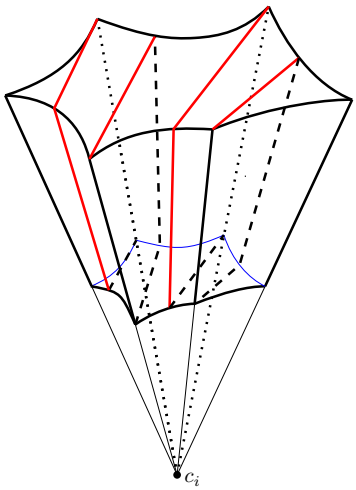


Figure 2 Vertical decomposition of a cell in the spherical coordinate system.

We decompose the cells τ of $\mathcal{A}_k^{(i)}(\mathcal{F})$ by constructing "vertical decompositions" of the corresponding 3D projections $\tau^\downarrow$ in the spherical coordinate system around c_i. As for standard vertical decompositions in 3D, we proceed in two stages. The first stage erects a wall from each edge of τ as follows. Fix an edge e of $\tau^\downarrow$. For a point $x \in e$, let $\lambda(x)$ be the ray emanating from c_i and passing through x, and let $h(x) = \lambda(x) \cap \tau^\downarrow$. By Lemma 5, $h(x)$ is a (connected) segment that touches e at one of its endpoints – it is either the top endpoint, for all points on e, or the bottom endpoint (with respect to distances from c_i). We erect the wall $\hat{e} = \bigcup_{x \in e} h(x)$ in $\tau^\downarrow$; parts of the wall may extend to infinity when $\tau^\downarrow$ is unbounded and e is a bottom edge. We repeat this step for all edges of $\tau^\downarrow$. These walls decompose $\tau^\downarrow$ into *frustums* (truncated cones), each of which, denoted by Δ, has a unique pair of *front* and *back* faces; the other faces of Δ lie on the created walls. (See Fig. 2.) The projections of the front and back faces of Δ on $\mathbb{S}^2$ (with c_i as the origin) are identical, and we denote this identical

projection by $\Delta^{\downarrow}$. The complexity of $\Delta^{\downarrow}$ may be arbitrarily large (a typical situation in the first stage of vertical decompositions in general; see, e.g., [19]). The subcell Δ itself is of the form $\Delta = \{(u, r) \mid u \in \Delta^{\downarrow}, r \in [g^-(u), g^+(u)]\}$, where g^-, g^+ are the functions of $G^{(i)}$ whose graphs contain the front and back faces of Δ, respectively.

The second stage decomposes Δ into constant-complexity frustums, which we refer to as *pseudo-cones*. We decompose $\Delta^{\downarrow}$ into spherical pseudo-trapezoids, by drawing portions of meridians within $\Delta^{\downarrow}$ from each vertex and meridian-tangency point. We then lift each pseudo-trapezoid $\sigma^{\downarrow}$ to form the pseudo-cone $\sigma = \{(u, r) \mid u \in \sigma^{\downarrow}, r \in [g^-(u), g^+(u)]\}$ in $\mathbb{R}^3$.

Repeating this step for all frustums created in the first stage, we obtain a decomposition of $\tau^{\downarrow}$ into pseudo-cones, each being a semi-algebraic set of constant complexity and bounded by up to six facets (compare with standard vertical decompositions in 3D [19]). We refer to these pseudo-cones as *elementary cells*. We note that each pseudo-cone ϕ is *defined* by a subset D_ϕ of at most seven functions of $\mathcal{F}$ (the function f_i and up to six additional functions that define the functions g_{ij} that form the frustum), in the sense that ϕ appears in the vertical decomposition of a 3D cell of $\mathcal{A}(D_\phi)$. By repeating this step for all cells of $\mathcal{M}_k^i$ and for all homothets K_i of $\mathcal{K}$, we obtain a decomposition of $\mathcal{M}_k = \mathcal{M}_k(\mathcal{K})$ into elementary cells. Finally, we lift each elementary cell of this decomposition vertically to $\mathcal{A}_k(\mathcal{F})$ in a straightforward manner, to obtain a corresponding decomposition of $\mathcal{A}_k(\mathcal{F})$.

Kaplan *et al.* [26, Lemma 6.2] have shown that the complexity of the vertical decomposition of the $(\leq k)$-level $\mathcal{A}_{\leq k}(\mathcal{G})$ of a set of $\mathcal{G}$ of m bivariate semi-algebraic functions of constant complexity is $O(k^2\psi(n/k)\lambda_s(k))$, where $\psi(r)$ is the maximum complexity of the lower envelope of a subset of $\mathcal{G}$ of size at most r, $\lambda_s(t)$ is the maximum length of Davenport-Schnizel sequences of order s composed of t symbols [34], and s is a constant depending on the complexity of the functions in $\mathcal{G}$. By applying the argument in [26] to our spherical coordinate system and using the worst-case bound $\psi(m) = O^*(m^2)$ on the complexity of the lower envelope of m constant-complexity bivariate functions [34], we conclude that the cells of $\mathcal{M}_k^i$ can be decomposed into $O^*(n^2k)$ elementary cells. Hence, $\mathcal{A}_k(\mathcal{F})$, or $\mathcal{M}_k(\mathcal{K})$, can be decomposed into $O^*(n^3k)$ elementary cells. By adapting the randomized incremental algorithm described in [26], $\mathcal{M}_k^i$, for each i, can be computed in $O^*(n^2k)$ expected time. We thus obtain:

▶ **Theorem 7.** *Let K be a compact, centrally-symmetric strictly convex semi-algebraic set in $\mathbb{R}^3$ of constant complexity. Let $\mathcal{K}$ be a set of n homothetic copies of K. The cells of $\mathcal{M}_k(\mathcal{K})$, and the cells of the k-level in the arrangement of their K-distance functions, can be decomposed into $O^*(n^3k)$ elementary cells, in $O^*(n^3k)$ expected time.*

3 Vertical Shallow Cuttings

A key notion that we need for constructing the dynamic NN data structures of Section 4 is that of a vertical shallow cutting of $\mathcal{A}(\mathcal{F})$, as studied in [15, 18, 26] (for simpler scenarios). For a parameter k, a *vertical k-shallow cutting* of $\mathcal{F}$ is a collection of pairwise openly disjoint semi-unbounded *pseudo-prisms* (prisms for short), where each prism consists of all points that lie vertically below some pseudo-cone, of the decomposition (of Section 2) that covers $\mathcal{A}_{\leq k}(\mathcal{F})$, and is thus a semi-algebraic set of constant complexity. Furthermore, the ceilings of these prisms collectively form an x_4-monotone surface that lies between $\mathcal{A}_k(\mathcal{F})$ and $\mathcal{A}_{2k}(\mathcal{F})$, and each prism is crossed by the graphs of at most $O(k)$ functions of $\mathcal{F}$. The *conflict list* of a prism σ is the set of functions whose graphs cross σ.

Following the ideas in [26], for a parameter $t \in [1, n]$, we construct a vertical (n/t)-shallow cutting as follows. We set two parameters: $r = \beta t \log t$, where β is a sufficiently large constant, independent of t, and $q = \beta \log t = r/t$. We choose a random subset $\mathsf{N} \subseteq \mathcal{F}$ consisting of

r functions, construct $\mathcal{A}_q(\mathsf{N})$, and compute the decomposition of $\mathcal{A}_q(\mathsf{N})$ into a family Ξ of pseudo-cones as described in Section 2. For each pseudo-cone $\tau \in \Xi$, we associate a label $\varphi(\tau)$ which is the function of N whose graph contains τ. Finally, we erect a semi-unbounded prism $\tau^{\uparrow}$ from each pseudo-cone τ of Ξ, given by $\tau^{\uparrow} = \bigcup\{\{c\} \times (-\infty, \rho] \mid (c, \rho) \in \tau\}$. Let $\Xi^{\uparrow}$ be the resulting set of pseudo-prisms. By Theorem 7, with large probability, $|\Xi| = O^*(r^3 q) = O^*(t^3)$. We next show that Ξ is a vertical k-shallow cutting of $\mathcal{F}$, where $k = n/t$, with high probability.

Range spaces and shallow ε-nets. Let $\Sigma = (\mathsf{X}, \mathcal{R})$ be a (finite) range space, where X is a set of objects and $\mathcal{R} \subseteq 2^{\mathsf{X}}$ a set of ranges. Let $0 < \varepsilon < 1$ be a given parameter. A subset $\mathsf{N} \subseteq \mathsf{X}$ is called a *shallow ε-net* of Σ if it satisfies the following two properties for every range $R \in \mathcal{R}$ and for any parameter $\zeta \geq 0$:

(i) If $|R \cap \mathsf{N}| \leq \zeta \log \frac{1}{\varepsilon}$ then $|R \cap \mathsf{X}| \leq \alpha(\zeta + 1)\varepsilon|\mathsf{X}|$, and

(ii) If $|R \cap \mathsf{X}| \leq \zeta\varepsilon|\mathsf{X}|$ then $|R \cap \mathsf{N}| \leq \alpha(\zeta + 1) \log \frac{1}{\varepsilon}$.

Here α is a suitable constant. Note the difference between shallow and standard ε-nets: Property (i) (with $\zeta = 0$) implies that a shallow ε-net is also a standard ε-net (possibly with a recalibration of ε). Property (ii) has no parallel in the case of standard ε-nets – there is no guarantee how a standard net interacts with small ranges (of the entire set X). The following result by Sharir and Shaul [35] is a generalization of the result on standard ε-nets [24].

▶ **Lemma 8** (Theorem 2.2 of Sharir and Shaul [35]). *Let $\Sigma = (\mathsf{X}, \mathcal{R})$ be a range space with VC-dimension d. With a suitable choice of the constant of proportionality, a random sample $\mathsf{N} \subseteq \mathsf{X}$ of $O\left(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon} + \log\frac{1}{\delta}\right)$ is a shallow ε-net with probability at least $1 - \delta$.*

Ξ is a vertical shallow cutting. We apply the above result to the range space $\Sigma = (\mathcal{F}, \mathcal{R})$, so that each range in $\mathcal{R}$ is the subset of the surfaces of $\mathcal{F}$ that cross some region in $\mathbb{R}^4$, taken from some family Γ of regions. Concretely, this includes the following families Γ: (i) the set of pseudo-cones generated by the decomposition of a set of at most seven functions of $\mathcal{F}$ (recall that a pseudo-cone is defined by at most seven functions), (ii) the set of pseudo-prisms erected on these pseudo-cones, (iii) the set of edges in an arrangement of five functions of $\mathcal{F}$ (an edge in the arrangement of a subset of $\mathcal{F}$ is defined by at most five functions), and (iv) the set of rays in the positive x_4-direction. Using standard arguments, it can be shown that the VC-dimension of Σ is finite. Therefore, by applying Lemma 8 with $\varepsilon = 1/t$ and $\delta = 1/3$ and choosing the constant of proportionality appropriately, the random subset $\mathcal{R}$ is a $(1/t)$-shallow net of Σ with probability at least $2/3$, so we assume that N is indeed a $(1/t)$-shallow net of Σ. Recall that $q = \beta \log t$. Assuming $\beta \geq 2\alpha$, where α is the constant in the definition of shallow nets, the converse of property (ii) of shallow nets, with $\zeta = 1$, implies that the level of any point p on $\mathcal{A}_q(\mathsf{N})$ with respect to $\mathcal{F}$ is at least n/t. Also, by property (i), the level of p is at most $O(n/t)$. Finally, the shallow net property also implies that the size of the conflict-list of any prism in $\Xi^{\uparrow}$ is $O(n/t)$. The conflict list of all prisms can be computed in $O^*(nt^3)$ time. Hence, we obtain:

▶ **Theorem 9.** *Let K be a compact, centrally-symmetric strictly convex semi-algebraic set in $\mathbb{R}^3$ of constant complexity. Let $\mathcal{K}$ be a set of n homothetic copies of K. For any parameter $t \in [1, n-1]$, there exists a vertical (n/t)-shallow cutting Ξ of the arrangement of the distance functions of $\mathcal{K}$ of size $O^*(t^3)$. The cutting Ξ, along with the conflict list of each of its prisms, can be computed in $O^*(nt^3)$ expected time.*

4 Dynamic Data Structures for Proximity Queries

In this section we describe a dynamic data structure for answering *intersection-detection* queries amid $\mathcal{K}$. Namely, for a query homothet $K_0 \in \mathbb{K}$, determine whether K_0 intersects some homothet of $\mathcal{K}$, and, if the answer is yes, return such a homothet. In addition to answering queries, the data structure can be updated efficiently as a homothet $K_0 \in \mathbb{K}$ is inserted into $\mathcal{K}$ or deleted from $\mathcal{K}$. The data structure can be extended to answering *intersection-reporting* queries, namely, reporting all homothets of $\mathcal{K}$ that intersect K_0, at an additional cost of $O(k)$.

As already noted, the insertion of a new homothet into $\mathcal{K}$ can be handled using the standard dynamization technique by Bentley and Saxe [13] (see also [9]). That is, assume that we have an intersection-detection data structure that can handle deletions, which can be constructed in $P(n)$ time, so that a query can be answered in $Q(n)$ time and an object can be deleted in $D(n)$ time. Then the amortized insertion time is $O((P(n)/n)\log n)$, the deletion time remains $D(n)$, and the overall query time is $O(Q(n))$, assuming that $Q(n) = \Omega(n^{\varepsilon})$.

We first describe a data structure that answers a query in $O^*(1)$ time using $O^*(n^3)$ space and handles a deletion in $O^*(n^2)$ amortized expected time, and then describe a linear-size data structure that answers a query in $O^*(n^{2/3})$ time and handles a deletion in $O(\log^2 n)$ amortized expected time.

4.1 Fast query-time data structure

We follow the same approach of Agarwal and Matoušek for performing dynamic halfspace range reporting [9] (slightly improved but more involved approaches were proposed in [15, 26], but the one in [9] will do for our setting).

Data structure. Let $\mathcal{F}$ be the collection of distance functions corresponding to the elements of $\mathcal{K}$. Roughly speaking, we build a tree data structure on $\mathcal{F}$ using vertical shallow cuttings. That is, we start with $\mathcal{F}$, choose a parameter $t > 1$, compute an (n/t)-vertical shallow cutting Ξ of $\mathcal{F}$ of size $O^*(t^3)$, in $O^*(nt^3)$ time, create a child for each prism τ of Ξ, recursively construct the data structure for the conflict list $\mathcal{F}_\tau$ of every $\tau \in \Xi$, and attach it as the subtree of the child corresponding to τ. The recursion proceeds until we reach cells τ with $|\mathcal{F}_\tau| = O(1)$, in which case we just store $\mathcal{F}_\tau$ at τ, as a list, say.

A query, with a copy $K_0 = (c_0, \rho_0)$ of K, is answered in a standard way, by traversing the tree in a top-down manner. At each node (prism) τ that we visit, we know that $(c_0, \rho_0) \in \tau$. We check whether (c_0, ρ_0) lies in any child prism of τ, and, if so, recurse at that child. Otherwise, we locate the child prism τ' for which (c_0, ρ_0) lies above τ', and report any element whose function belongs to the conflict list $\mathcal{F}_{\tau'}$.

However, this simple recursive approach runs into a technical difficulty when we delete a function from $\mathcal{F}$. When deleting a function from the root, the deletion has to be propagated to some of its children, to their children, and so on. On average, each function appears in the conflict lists of $O^*(t^2)$ prisms of Ξ. When such a "good" function is deleted, the deletion is propagated to only $O^*(t^2)$ descendants, leading to $O^*(n^2)$ overall deletion time, as can easily be verified. However, some of the functions (in fact, up to n/t of them) may appear in the conflict lists of $\Omega(t^3)$ prisms of Ξ. If these "bad" functions are the first n/t functions to be deleted and then the data structure is reconstructed, the overall cost of the deletion operations will be too high. As in [9, 26], we circumvent this difficulty by maintaining a partition of $\mathcal{F}$ into a few subsets and constructing a cutting for each subset that is good for all functions in that subset (i.e., each function in the subset appears in only $O^*(t^2)$ conflict lists).

We now describe the data structure in detail. We fix a sufficiently large constant $t > 0$. The data structure is periodically reconstructed after performing some deletions. We use m to denote the size of $\mathcal{F}$ when the data structure was previously reconstructed and n to denote its current size. We reconstruct the data structure after deleting $m/2t$ functions from $\mathcal{F}$. Thus $n \geq m(1 - 1/2t)$. Let $P(r)$ denote the maximum time spent in constructing the data structure for a set of r functions. We pay for the reconstruction cost by charging $2tP(m - m/2t)/m \leq 2tP(n)/n$ time to each of the $m/2t$ delete operations that occurred before the reconstruction (the inequality holds because P is assumed to be superlinear). The data structure for $\mathcal{F}$ is a (recursively defined) tree $\Psi(\mathcal{F})$. We describe how a subtree $\Psi(\mathcal{G})$, for a subset $\mathcal{G}$ of ν functions, is constructed.

If $\nu \leq n_0$, for some sufficiently large constant n_0, then $\Psi(\mathcal{G})$ is a single leaf, and we simply store $\mathcal{G}$ at that leaf. So assume that $\nu > n_0$. The root of $\Psi(\mathcal{G})$ stores the following data:

- A partition of $\mathcal{G}$ into subsets $\mathcal{G}_1, \ldots, \mathcal{G}_u$, $u \leq \lceil \log_2 t \rceil$, as described below.
- For each $1 \leq i \leq u$, a vertical (ν/t)-shallow cutting Ξ_i for $\mathcal{G}_i$. For each $\Delta \in \Xi_i$, we store its conflict list $\mathcal{G}_{i,\Delta}$.
- For each $1 \leq i \leq u$ and $\Delta \in \Xi_i$, a pointer to the tree $\Psi(\mathcal{G}_{i,\Delta})$.
- For each function $f \in \mathcal{G}_i$, the set $L_f \subseteq \Xi_i$ of prisms Δ crossed by the graph of f.
- A counter χ that keeps track of how many functions of $\mathcal{G}$ can be deleted before we reconstruct $\Psi(\mathcal{G})$.

After the construction of $\Psi(\mathcal{G})$, before any deletions occur, the following properties hold:

(P1) The counter χ is set to $\nu/2t$.
(P2) For each i, the x_4-monotone surface formed by the pseudo-cones of Ξ_i (i.e., the ceilings of prisms of Ξ_i) lie above $\mathcal{A}_{\nu/t}(\mathcal{G}_i)$.
(P3) For every i and $\Delta \in \Xi_i$, $|\mathcal{G}_{i,\Delta}| \leq cn/t$, for some suitable absolute constant c.
(P4) Each function $f \in \mathcal{G}_i$ appears in the conflict lists of at most $\kappa := 2C_1 t^{2+\delta}$ prisms of Ξ_i, where C_1 and $\delta > 0$ are the constants appearing in (1) below.

We now describe the construction of the partition $\mathcal{G}_1, \ldots, \mathcal{G}_u$, which proceeds iteratively. Suppose we have constructed $\mathcal{G}_1, \ldots, \mathcal{G}_{i-1}$. Set $\mathcal{G}_i^* = \mathcal{G} \setminus \bigcup_{j<i} \mathcal{G}_i$ and put $\nu_i = |\mathcal{G}_i^*|$. If $\nu_i \leq \nu/t$, we stop. Otherwise, set $t_i = t\nu_i/\nu$ and $k = \nu_i/t_i = \nu/t$.

We construct a vertical k-shallow cutting Ξ_i of $\mathcal{G}_i^*$ of size $O(t_i^{3+\delta})$, for an arbitrarily small constant $\delta > 0$. For each $\Delta \in \Xi_i$, $|\mathcal{G}_{i,\Delta}^*| \leq c\nu_i/t_i = c\nu/t$. We note that

$$\sum_{\Delta \in \Xi_i} |\mathcal{G}_{i,\Delta}^*| \leq O(t_i^{3+\delta}) \frac{c\nu}{t} \leq C_1 \left(\frac{t\nu_i}{\nu}\right)^{3+\delta} \frac{\nu}{t} \leq C_1 \nu_i t^{2+\delta}. \tag{1}$$

A function $f \in \mathcal{G}_i^*$ is called *good* if it appears in the conflict list of at most κ prisms of Ξ_i and *bad* otherwise. Let $\mathcal{G}_i$ be the set of good functions of $\mathcal{G}_i^*$; set $\mathcal{G}_{i+1}^* = \mathcal{G}_i^* \setminus \mathcal{G}_i$. By (1), $|\mathcal{G}_{i+1}^*| \leq \nu_i/2$. Hence, $u \leq \lceil \log_2 t \rceil$. It is easily seen that properties (P1)–(P4) hold after the construction. This completes the description of the data structure. The total size and preprocessing time are $O^*(n^3)$.

Query procedure. Let $K(c_0, \rho_0) \in \mathbb{K}$ be a query homothet. Recall that $K(c_0, \rho_0)$ intersects a homothet of $\mathcal{K}$ if the point (c_0, ρ_0) lies above the lower envelope of the corresponding set $\mathcal{F}$ of functions. The query is performed as follows. We search $\Psi(\mathcal{F})$ in a top-down manner with (c_0, ρ_0). Suppose we are at the root of a subtree $\Psi(\mathcal{G})$. If $\Psi(\mathcal{G})$ is a leaf, we check all functions of $\mathcal{G}$ in a brute-force manner and return an answer in $O(n_0) = O(1)$ time. Otherwise, for

each $i \leq u$, we check whether (c_0, ρ_0) lies above the lower envelope of $\mathcal{G}_i$. If the answer is yes for any i, we return yes and also return an intersecting homothet (see below). Otherwise, we return no.

For a fixed $i \leq u$, we proceed as follows. We search with c_0, in a brute force manner, in $O^*(t^3) = O(1)$ time, to determine the pseudo-cone $\tau^{\downarrow}$ in the projection of Ξ_i that contains c_0. Let f_i be the function associated with τ, i.e., the ceiling of Δ lies in the graph of f_i. We test whether $\rho_0 \geq f_i(c_0)$. If so, K_i and $K(c_0, \rho_0)$ intersect, so we return yes and report K_i as an intersection witness and terminate the query. If not, (c_0, ρ_0) lies in the semi-unbounded prism Δ, and we continue the search recursively in the data structure $\Psi(\mathcal{G}_{i,\Delta})$.

Let $Q(\nu)$ be the maximum time spent by the query procedure in the subtree $\Psi(\mathcal{G})$ storing at most ν functions. Then we obtain the following recurrence:

$$Q(\nu) \leq \lceil \log_2 t \rceil Q(c\nu/t) + O(t^{3+\delta}).$$

Its solution is easily seen to be $Q(\nu) = O(n^{\varepsilon})$, for any constant $\varepsilon > 0$, provided that n_0 and t are chosen sufficiently large.

Deletion procedure. Let $f \in \mathcal{F}$ be a function that we wish to delete from $\mathcal{F}$. Again, we visit $\Psi(\mathcal{F})$ in a top-down manner. Suppose we are at the root v of a subtree $\Psi(\mathcal{G})$. If v is a leaf, then we simply delete f from $\mathcal{G}$ and stop. If v is an internal node, we first decrement the counter χ stored at v. If χ becomes 0, we reconstruct $\Psi(\mathcal{G})$ from scratch (for the current $\mathcal{G}$). Otherwise, we find the index i such that $f \in \mathcal{G}_i$. For each $\Delta \in L_f$, we delete f from $\mathcal{G}_{i,\Delta}$ and then recursively delete f from $\Psi(\mathcal{G}_{i,\Delta})$. By construction, $|L_f| \leq \kappa$. The deletion procedure maintains the properties (P3) and (P4), and (P2) is replaced with a slightly weaker property:

(P2') For every i, the x_4-monotone surface formed by the pseudo-cones of Ξ_i (i.e., the ceilings of prisms of Ξ_i) lie above or on $\mathcal{A}_{\nu/2t}(\mathcal{G}_i)$.

The correctness of the query procedure follows from property (P2'). Following the analysis in [9], the amortized deletion time, including the time spent in reconstructing the data structure, is $O^*(n^2)$. Putting everything together, and combining this with the technique of Bentley and Saxe [13] for insertions, we obtain:

▶ **Lemma 10.** *$\mathcal{K}$ can be preprocessed, in $O^*(n^3)$ expected time, into a data structure of size $O^*(n^3)$, so that an intersection-detection query, including the cost of reporting an intersecting member of $\mathcal{K}$ if one exists, can be performed in $O^*(1)$ time. A homothet can be inserted into $\mathcal{K}$ or deleted from $\mathcal{K}$ in $O^*(n^2)$ amortized expected time.*

4.2 Linear-size data structure

Next, we present a linear-size dynamic data structure that supports intersection-detection queries in $O^*(n^{2/3})$ time each, and can handle updates (insertions and deletions) in $O(\log^2 n)$ amortized expected time per update.

For a homothet $K(c_0, \rho_0)$, let $f^+_{c_0,\rho_0} = \{(c, \rho) \in \mathbb{R} \mid \rho \geq f_{c_0,\rho_0}(c)\}$ be the region lying above the graph of the distance function f_{c_0,ρ_0}, which corresponds to the set of homothets that intersect $K(c_0, \rho_0)$. Put $\mathcal{K}^* = \{(c, \rho) \mid K(c, \rho) \in \mathcal{K}\} \subset \mathbb{R}^4$. For a query homothet $K(c_0, \rho_0)$, we wish to determine whether any point of $\mathcal{K}^*$ lies in $f^+_{c_0,\rho_0}$. We construct a linear-size partition tree for answering these queries, following the same approach as in [9, 32, 35]. We need a few definitions. Let $\mathbb{F}$ be the set of distance functions corresponding to the homothets in $\mathbb{K}$. Let $P \subset \mathbb{K}$ be a set of n points (representing homothets of K). For a parameter

$k \geq 1$, we call a semi-algebraic set $\gamma \subset \mathbb{K}$, which is semi-unbounded in the $(-x_4)$-direction, *k-shallow* if $|P \cap \gamma| \leq k$. A major ingredient of the approach in [32, 35] is to construct a so-called *test set* composed of a small number of semi-algebraic sets, which represent well (in the sense made precise below) all distance functions of $\mathbb{F}$ that are (n/r)-shallow, for a given parameter $r > 1$, with respect to P. Formally, a finite collection Φ of semi-algebraic sets of constant complexity (not necessarily a subset of $\mathbb{F}$) is called a *test set* for (n/r)-shallow ranges in $\mathbb{F}$ with respect to P if it satisfies the following properties:

- **(i)** Every set in Φ is (n/r)-shallow with respect to P.
- **(ii)** The complement of the union of any m sets of Φ can be decomposed into at most $\zeta(m)$ "elementary cells" (semi-algebraic sets of constant complexity) for any $m \geq 1$, where $\zeta(m)$ is a suitable monotone increasing superlinear function of m.
- **(iii)** Any (n/r)-shallow set $\sigma \in \mathbb{F}$ can be covered by the union of at most δ ranges of Φ, where δ is a constant (independent of r).

Sharir and Shaul [35] showed that if there exists a test set of size $r^{O(1)}$ for (n/r)-shallow ranges of $\mathbb{F}$, then one can construct a linear-size partition tree on P so that an $\mathbb{F}$-emptiness query can be answered in $O^*(n/\zeta^{-1}(n))$ time, for the corresponding function ζ.

Test-set construction. We describe an algorithm for constructing a test set of size $O(r^4)$ with $\zeta(m) = O^*(m^3)$ and $\delta = 1$. Let $\mathcal{F}$ be the set of distance functions of $\mathcal{K}$ as above. We take a random subset $R \subseteq \mathcal{F}$ of $s = cr \log r$ functions, for some sufficiently large constant r, where $c > 0$ is another sufficiently large absolute constant, independent of, and much smaller than r, and construct the (standard) 4D vertical decomposition $\mathcal{A}^{\shortparallel}(R)$ of the arrangement $\mathcal{A}(R)$, of size $O^*(r^4)$, as described in [3,30,34]. By a standard random-sampling argument [24], with probability at least $1/2$, each cell of $\mathcal{A}^{\shortparallel}(R)$ is crossed by at most n/r functions of $\mathcal{F}$, provided that c is chosen sufficiently large. If this is not the case, we discard R and choose another random subset, until we find, in expected $O(1)$ trials, one with the desired property. We clip $\mathcal{A}^{\shortparallel}(R)$ within the halfspace $x_4 \geq 0$ (to restrict it to within $\mathbb{K}$). We choose a subset Ξ of the cells of $\mathcal{A}^{\shortparallel}(R)$, consisting of those cells that have at most n/r functions of $\mathcal{F}$ passing *fully below* them. By construction, these cells cover $\mathcal{A}_{\leq n/r}(\mathcal{F})$, and are contained in $\mathcal{A}_{\leq 2n/r}(\mathcal{F})$.

Let τ be a cell of Ξ. Set $\Phi_\tau = \bigcup_{(c_0,\rho_0)\in\tau} f^+_{c_0,\rho_0}$. Since τ and f_{c_0,ρ_0} are semi-algebraic sets of constant complexity, Φ_τ is also a semi-algebraic set of constant complexity. For a homothet $K_i \in \mathcal{K}$, the point $K_i^* \in \mathbb{K}$ lies in Φ_τ if and only if there is a point $(c_0, \rho_0) \in \tau$ such that $K_i^* \in f^+_{c_0,\rho_0}$. This happens when $(c_0, \rho_0) \in f_i^+$, i.e., the graph of the function f_i crosses τ or lies below τ. By construction, there are at most $n/r + n/r = 2n/r$ such functions. Consequently, Φ_τ is $(2n/r)$-shallow with respect to the points of $\mathcal{K}^*$ with $\delta = 1$.

Set $\Phi = \{\Phi_\tau \mid \tau \in \Xi\}$. Φ is a family of $O^*(r^4)$ constant-complexity semi-algebraic sets in $\mathbb{K}$, each of which is $(2n/r)$-shallow with respect to $\mathcal{K}^*$. This is the desired test set. Each set Φ_τ is unbounded in the positive x_4-direction, therefore the complement of the union of a subset of m sets is the region lying below their lower envelope. By a result in [5], the complement of their union (i.e., the region below their lower envelope) can be decomposed into $O^*(m^3)$ pseudo-prisms. Hence, we obtain:

▶ **Lemma 11.** *Let $P \subset \mathbb{K}$ be a set of n points and $r \geq 1$ a parameter. A test set Φ of size $O(r^4)$ for the functions in $\mathbb{F}$ that are (n/r)-shallow with respect to P can be computed in $O^*(r^4)$ expected time, so that $\zeta(m) = O^*(m^3)$ and $\delta = 1$.*

Using Lemma 11 and adapting the approach in [35], we can preprocess $\mathcal{K}$ into a linear-size data structure that supports intersection-detection queries in $O^*(n^{2/3})$ time. By (re)constructing portions of the partition tree periodically (as in the previous data structure), deletions can be handled efficiently. Omitting the straightforward details, we obtain:

▶ **Lemma 12.** *Let $\mathcal{K}$ be a set of n homothets of K. $\mathcal{K}$ can be preprocessed, in $O(n \log n)$ expected time, into a data structure of size $O(n)$, so that an intersection-detection query can be answered in $O^*(n^{2/3})$ time. A homothet of K can be inserted into or deleted from the data structure in $O(\log^2 n)$ amortized time.*

By combining this data structure with Lemma 10 in a standard manner [1], we can obtain the following space/query-time trade-off:

▶ **Theorem 13.** *Let $\mathcal{K}$ be a set of n homothets of K, and let $s \in [n, n^3]$ be a storage parameter. $\mathcal{K}$ can be preprocessed, in $O^*(s)$ expected time, into a data structure of size $O^*(s)$, so that an intersection-detection query can be answered in $O^*(n/s^{1/3})$ time. A homothet can be inserted into or deleted from the data structure in $O^*(s/n)$ amortized time.*

Finally, by combining this data structure with the parametric-search technique [8], we establish Theorem 1 (for answering nearest-neighbor queries in $\mathcal{K}$ under the K-distance).

References

1 P. K. Agarwal, B. Aronov, E. Ezra, M. J. Katz, and M. Sharir. Intersection queries for flat semi-algebraic objects in three dimensions and related problems. *ACM Transactions on Algorithms*, 21:25:1–25:59, 2025. doi:10.1145/3721290.

2 P. K. Agarwal, B. Aronov, E. Ezra, and J. Zahl. An efficient algorithm for generalized polynomial partitioning and its applications. *SIAM Journal on Computing*, 50:760–787, 2021. doi:10.1137/19M1268550.

3 P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM Journal on Computing*, 29(3):912–953, 1999. doi:10.1137/S0097539795295936.

4 P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. American Mathematical Society, Providence, RI, 1999. doi:10.1090/conm/223/03131.

5 P. K. Agarwal, E. Ezra, and M. Sharir. Vertical decomposition in 3D and 4D with applications to line nearest-neighbor searching in 3D. In *35th ACM-SIAM Symposium on Discrete Algorithms (SODA 2024)*, pages 150–170, 2024. doi:10.1137/1.9781611977912.8.

6 P. K. Agarwal, H. Kaplan, M. J. Katz, and M. Sharir. Segment proximity graphs and nearest neighbor queries amid disjoint segments. In *32nd European Symposium on Algorithms (ESA 2024)*, pages 7:1–7:20, 2024. doi:10.4230/LIPIcs.ESA.2024.7.

7 P. K. Agarwal, M. J. Katz, and M. Sharir. On reverse shortest paths in geometric proximity graphs. *Computational Geometry*, 117:102053, 2024. doi:10.1016/j.comgeo.2023.102053.

8 P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM Journal on Computing*, 22:794–806, 1993. doi:10.1137/0222051.

9 P. K. Agarwal and J. Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13:325–345, 1995. doi:10.1007/BF01293483.

10 A. Andoni and P. Indyk. Nearest neighbors in high dimensional spaces. In J. E. Goodman, J. O'Rourke, and C. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, Boca Raton, FL, 3rd edition, 2017. Chapter 43. doi:10.1201/9781315119601.

11 R. Ben Avraham, O. Filtser, H. Kaplan, M. J. Katz, and M. Sharir. The discrete and semicontinuous Fréchet distance with shortcuts via approximate distance counting and selection. *ACM Transactions on Algorithms*, 11(4):29:1–29:29, 2015. doi:10.1145/2700222.

12 S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Springer, Berlin, 2nd edition, 2006.

13 J. L. Bentley and J. B. Saxe. Decomposable searching problems I: Static-to-dynamic transformation. *Journal of Algorithms*, 1:301–358, 1980. doi:10.1016/0196-6774(80)90015-2.

14 S. Cabello and M. Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry*, 48:360–367, 2015. doi:10.1016/j.comgeo.2014.12.003.

15 T. M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest-neighbor queries. *Journal of the ACM*, 57:16:1–16:15, 2010. doi:10.1145/1706591.1706596.

16 T. M. Chan. Dynamic generalized closest pair: Revisiting Eppstein's technique. In *3rd Symposium on Simplicity in Algorithms (SOSA 2020)*, pages 33–37, 2020. doi:10.1137/1.9781611976014.6.

17 T. M. Chan and Z. Huang. Faster algorithms for reverse shortest path in unit-disk graphs and related geometric optimization problems: Improving the shrink-and-bifurcate technique. In *41st International Symposium on Computational Geometry (SoCG 2025)*, pages 32:1–32:14, 2025. doi:10.4230/LIPIcs.SoCG.2025.32.

18 T. M. Chan and K. Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete & Computational Geometry*, 56:866–881, 2016. doi:10.1007/S00454-016-9784-4.

19 B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. In *16th International Colloquium on Automata, Languages and Programming (ICALP 1989)*, pages 179–193, 1989. Also in Theoretical Computer Science 84 (1991), 77–105. doi:10.1007/BFB0035760.

20 K. L. Clarkson. Nearest neighbor searching and metric space dimensions. In G. Shakhnarovich, T. Darrell, and P. Indyk, editors, *Nearest-Neighbor Methods in Learning and Vision*. MIT Press, 2006.

21 K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989. doi:10.1007/BF02187740.

22 M. de Berg and S. Cabello. An $O(n \log n)$ algorithm for single-source shortest paths in disk graphs. In *33rd European Symposium on Algorithms (ESA 2025)*, pages 81:1–81:15, 2025. doi:10.4230/LIPIcs.ESA.2025.81.

23 D. Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. *ACM Journal of Experimental Algorithmics*, 5:1, 2000. doi:10.1145/351827.351829.

24 D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987. doi:10.1007/BF02187876.

25 H. Kaplan, M. J. Katz, R. Saban, and M. Sharir. The unweighted and weighted reverse shortest path problem for disk graphs. *ACM Transactions on Algorithms*, 2023. To appear. Also in 31st European Symposium on Algorithms (ESA 2023), 67:1–67:14.

26 H. Kaplan, W. Mulzer, L. Roditty, P. Seiferth, and M. Sharir. Dynamic planar voronoi diagrams for general distance functions and their algorithmic applications. *Discrete & Computational Geometry*, 64:838–904, 2020. doi:10.1007/s00454-020-00243-7.

27 M. J. Katz, R. Saban, and M. Sharir. Near-linear algorithms for visibility graphs over a 1.5-dimensional terrain. In *32nd European Symposium on Algorithms (ESA 2024)*, pages 77:1–77:17, 2024. doi:10.4230/LIPICS.ESA.2024.77.

28 M. J. Katz, R. Saban, and M. Sharir. BFS and reverse shortest paths for ball intersection graphs in three and higher dimensions. In *36th International Symposium on Algorithms and Computation (ISAAC 2025)*, pages 45:1–45:15, 2025. doi:10.4230/LIPICS.ISAAC.2025.45.

29 K. Klost. An algorithmic framework for the single source shortest path problem with applications to disk graphs. *Computational Geometry*, 111:101979, 2023. doi:10.1016/j.comgeo.2022.101979.

30 V. Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *Journal of the ACM*, 51:699–730, 2004. doi:10.1145/1017460.1017461.

31 C.-H. Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. *SIAM Journal on Computing*, 51(3):723–765, 2022. doi:10.1137/20M1388371.

32 J. Matoušek. Reporting points in halfspaces. *Computational Geometry*, 2:169–186, 1992. doi:10.1016/0925-7721(92)90006-E.

33 J. Matoušek and Z. Patáková. Multilevel polynomial partitioning and simplified range searching. *Discrete & Computational Geometry*, 54:22–41, 2015. doi:10.1007/s00454-015-9701-2.

34 M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.

35 M. Sharir and H. Shaul. Semialgebraic range reporting and emptiness searching with applications. *SIAM Journal on Computing*, 40(4):1045–1074, 2011. doi:10.1137/090765092.

36 H. Wang and Y. Zhao. Reverse shortest path problem for unit-disk graphs. *Journal of Computational Geometry*, 14(1):14–47, 2023. doi:10.20382/JOCG.V14I1A2.

Bifunction and Interlevel Delaunay Trifiltrations

Ángel Javier Alonso
CUNEF Universidad, Madrid, Spain
Institute of Geometry, Graz University of Technology, Austria

Michael Kerber
Institute of Geometry, Graz University of Technology, Austria

Tung Lam
University at Albany, SUNY, NY, USA

Michael Lesnick
University at Albany, SUNY, NY, USA

Abhishek Rathod
Ben-Gurion University of the Negev, Be'er-Sheva, Israel

Abstract

A key property of the Delaunay filtration is that it is topologically (i.e., weakly) equivalent to the offset (union-of-balls) filtration. Recently, this filtration has been extended to point clouds equipped with an $\mathbb{R}$-valued function, yielding a computable 2-parameter filtration that satisfies an analogous weak equivalence. Motivated in part by the study of time-varying data, we introduce a 3-parameter extension of the Delaunay filtration for point clouds equipped with an $\mathbb{R}^2$-valued function, also satisfying an analogous weak equivalence. For a point cloud $X \subset \mathbb{R}^d$, our trifiltration has size $O\big(|X|^{\lceil (d+1)/2 \rceil + 1}\big)$. We present an algorithm that computes this trifiltration in time $O\big(|X|^{\lceil d/2 \rceil + 2}\big)$, together with an implementation. Our experiments demonstrate that the implementation can handle thousands of points in $\mathbb{R}^3$, with memory growth that is nearly linear.

2012 ACM Subject Classification Mathematics of computing → Topology; Theory of computation → Computational geometry

Keywords and phrases Delaunay triangulation, Multiparameter persistent homology, Interlevel, Bowyer-Watson

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.5

Related Version *Full Version*: https://doi.org/10.5281/zenodo.19346228

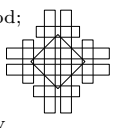
Supplementary Material *Software*: https://bitbucket.org/mkerber/function_delaunay [37]
archived at swh:1:dir:95fb3611456975ff368b3cfb5604f4bd9011264e
Dataset: https://doi.org/10.5281/zenodo.19227801

Funding *Ángel Javier Alonso*: Austrian Science Fund (FWF) grants 10.55776/P33765 and 10.55776/W1230.
Michael Kerber: Austrian Science Fund (FWF) grant 10.55776/P33765.
Tung Lam: Simons Foundation Award 96384.
Michael Lesnick: Simons Foundation Award 963845.
Abhishek Rathod: European Research Council (ERC) grant titled PARAPATH (101039913).

1 Introduction

Background. Given a finite point cloud $X \subset \mathbb{R}^d$, the *offset filtration* of X is the nested sequence of spaces $\mathcal{O}(X) = (\mathcal{O}(X)_r)_{r \in \mathbb{R}^+}$, where $\mathbb{R}^+ = [0, \infty)$ and $\mathcal{O}(X)_r$ is the union of closed balls of radius r centered the points of X (Figure 1). In topological data analysis (TDA), one often computes the persistent homology of $\mathcal{O}(X)$. However, directly computing $\mathcal{O}(X)$


© Ángel Javier Alonso, Michael Kerber, Tung Lam, Michael Lesnick, and Abhishek Rathod;
licensed under Creative Commons License CC-BY 4.0




is difficult, due to its non-combinatorial nature. Therefore, one usually instead computes the *Delaunay filtration* $\mathcal{D}(X)$ (also called the α-filtration) [30, 28] (Figure 1), a filtration of the Delaunay triangulation $\mathrm{Del}(X)$ of X that is topologically equivalent (i.e., weakly equivalent, see Definition 19) to $\mathcal{O}(X)$, and therefore has the same persistent homology. For low-dimensional Euclidean data, $\mathcal{D}(X)$ is the standard choice of filtration in TDA, because it is far more computationally efficient than alternatives like the Rips and Čech filtration. Bauer and Edelsbrunner [5] showed that $\mathcal{O}(X)$ is also weakly equivalent to a variant of $\mathcal{D}(X)$ called the *Delaunay-Čech* filtration and denoted $\mathcal{DC}(X)$, where the birth index of each simplex in the filtration is the radius of its minimum enclosing ball.

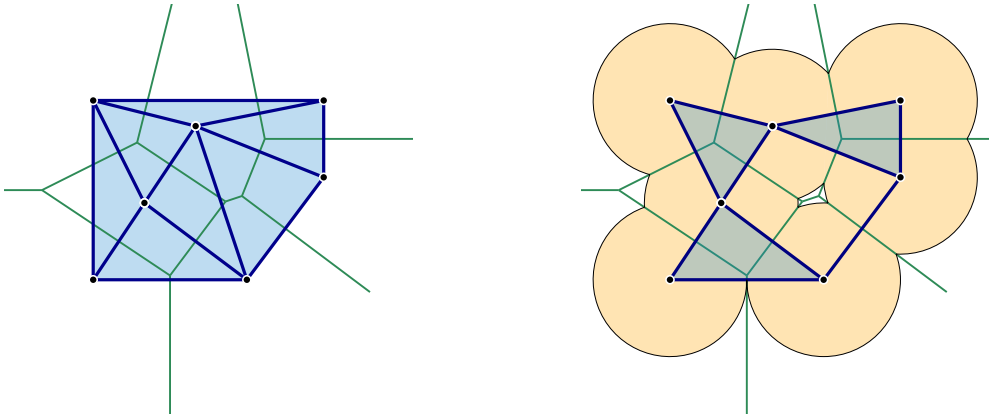


Figure 1 Left: For a point set $X \subset \mathbb{R}^2$, the Voronoi decomposition $\mathrm{Vor}(X)$ (green) and the corresponding Delaunay triangulation $\mathrm{Del}(X)$ (dark blue). Right: The offset $\mathcal{O}_r(X)$ (orange) and Delaunay complex $\mathcal{D}(X)_r \subset \mathrm{Del}(X)$ for a fixed radius r.

In many TDA applications, the point cloud X comes equipped with a function $\delta\colon X \to \mathbb{R}$ [20, 18]. It is then natural to seek a refinement of the persistent homology of X which is sensitive to the structure of this function. Functions considered in the TDA literature include (co)density, eccentricity (centrality), discrete curvature [18], partial charge [17, 16], and time [46]. To extend the persistence pipeline to point clouds equipped with multiple functions, Carlsson and Zomorodian [18] proposed *multiparameter persistence*, and specifically, the *sublevel offset bifiltration* $\mathcal{O}(\delta) = (\mathcal{O}(\delta)_{s,r})_{(s,r)\in\mathbb{R}\times\mathbb{R}^+}$, where

$$\mathcal{O}(\delta)_{s,r} = \mathcal{O}(\{x \in X \mid \delta(x) \leq s\})_r.$$

Sublevel Čech and sublevel offset bifiltrations are defined analogously, and can be computed in essentially the same way as the usual Rips and Čech filtrations. Defining an analogous sublevel Delaunay bifiltration is more subtle because of the non-monotonicity of Delaunay complexes with respect to insertion of points. However, recent work [3] introduced sublevel Delaunay and sublevel Delaunay-Čech bifiltrations that are both weakly equivalent to the sublevel offset bifiltration and amenable to efficient computation. Experiments with a publicly available implementation showed that, in practice, computing the sublevel Delaunay-Čech bifiltration is only modestly more expensive than computing the ordinary Delaunay filtration. The work [3] is part of a large body of recent work aiming to realize multiparameter persistence as a computationally viable data analysis tool; for an introduction, see [14].

Contributions. In this paper, we extend the recent results from [3] on sublevel-Delaunay bifiltrations to functions $\gamma\colon X \to \mathbb{R}^2$, where $X \subset \mathbb{R}^d$ is in general position. Specifically, we introduce the *sublevel Delaunay* and *sublevel Delaunay-Čech trifiltrations* of γ, denoted $\mathcal{D}(\gamma)$

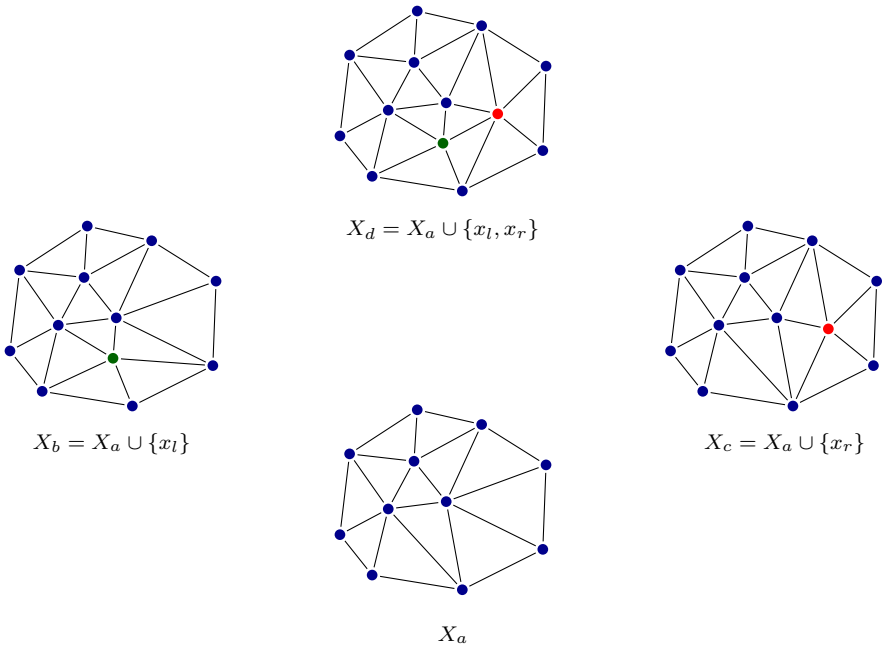


Figure 2 Non-monotonicity of Delaunay triangulations of planar point clouds X_a, $X_b = X_a \cup \{x_l\}$, $X_c = X_a \cup \{x_r\}$, and $X_d = X_a \cup \{x_l, x_r\}$. The points x_l and x_r are shown in green and red.

and $\mathcal{DC}(\gamma)$, respectively. We show that both trifiltrations have size $O(|X|^{\lceil (d+1)/2 \rceil + 1})$, can be computed in time $O(|X|^{\lceil d/2 \rceil + 2})$, and are weakly equivalent to the *sublevel offset trifiltration*, defined as follows:

▶ **Definition 1.** *The* sublevel offset trifiltration *of* γ*, denoted* $\mathcal{O}(\gamma)$*, is the collection of Euclidean subspaces* $\left(\mathcal{O}_{a,r}\right)_{(a,r)\in\mathbb{R}^2\times\mathbb{R}^+}$*, where* $\mathcal{O}_{a,r} \coloneqq \mathcal{O}(X_a)_r$ *and* $X_a \coloneqq \{x \in X \mid \gamma(x) \leq a\}$*. Here* $\leq$ *denotes the product partial order on* $\mathbb{R}^2$*.*

Our approach to defining and computing the trifiltrations $\mathcal{D}(\gamma)$, $\mathcal{DC}(\gamma)$ parallels the approach of [3] for $\mathbb{R}$-valued functions, but is necessarily more involved. To illustrate the problem, consider the four point clouds $X_a \subset X_b, X_c \subset X_d$ of Figure 2. While for any $r \in \mathbb{R}^+$, the offsets satisfy $\mathcal{O}(X_a)_r \subset \mathcal{O}(X_b)_r, \mathcal{O}(X_c)_r \subset \mathcal{O}(X_d)_r$, these containment relations do not hold for the corresponding Delaunay triangulations, nor for the Delaunay complexes of radius r. The construction of [3] applies when the point clouds are totally ordered by containment, but does not apply here because neither X_b nor X_c contains the other. In this paper, we adapt the approach of [3] to address this case.

As in the previous work on $\mathbb{R}$-valued functions, the most important part of our approach is the definition and computation of the maximum complex (i.e., colimit) of the trifiltrations $\mathcal{D}(\gamma)$ and $\mathcal{DC}(\gamma)$. This complex is the same for both trifiltrations; we call it the *incremental complex* and denote it as $\mathcal{I}$ (Definition 3). The complex $\mathcal{I}$ has vertex set X, and contains the union of the Delaunay triangulations of all sublevel sets of γ as a subcomplex. We give a characterization of $\mathcal{I}$ in terms of *conflict pairs* (see Definition 6), and use this to obtain a simple algorithm for computing $\mathcal{I}$, based on the standard Bowyer-Watson algorithm for incremental computation of Delaunay triangulations. We then introduce optimizations to this algorithm, exploiting the fact that Delaunay triangulations are *local*, in a sense made precise by Lemma 16.

For $\sigma \subset X$ non-empty, let $\gamma(\sigma) = \bigvee_{x\in\sigma} \gamma(x)$, where $\bigvee$ denotes the *join*, i.e., coordinate-wise maximum. Given $\mathcal{I}$, the trifiltration $\mathcal{D}(\gamma)$ is defined by assigning each simplex $\sigma \in \mathcal{I}$ the birth index $(\gamma(\sigma), \omega_\sigma)$, where ω_σ is the *incremental Delaunay radius*, which we will define in Definition 3. $\mathcal{DC}(\gamma)$ is defined by assigning each $\sigma \in \mathcal{I}$ the birth index $(\gamma(\sigma), m_\sigma)$, where m_σ is the radius of the minimal enclosing ball of σ. To show that $\mathcal{D}(\gamma)$, $\mathcal{DC}(\gamma)$, and $\mathcal{O}(\gamma)$ are topologically equivalent, we use an argument based on two-parameter mapping telescopes and functorial version of the nerve theorem [6]. We introduce a practical algorithm for computing $\mathcal{D}(\gamma)$ and $\mathcal{DC}(\gamma)$, and implement the algorithm to compute $\mathcal{DC}(\gamma)$. We perform experiments on point clouds in $\mathbb{R}^2$ and $\mathbb{R}^3$ with up to 16000 points. In these experiments, the memory usage scales nearly linearly, while run time scales nearly quadratically, suggesting that substantially larger computations are feasible. Our implementation is publicly available [37].

Motivation. While there is a substantial recent literature on computing bifiltrations of point cloud and metric data in TDA [23, 3, 9, 44, 2, 4], to the best of our knowledge, our work is the first to seriously engage with the algorithmic aspects of trifiltrations. We hope that our work will lower the barrier to practical data analysis with 3-parameter persistence, which has seen very limited use so far.

Our specific motivation for introducing computable Delaunay-type models of sublevel-offset trifiltrations is threefold: First, we wish to enable persistence analysis which simultaneously considers a pair of functions on a point cloud, e.g., density and eccentricity. Indeed, if one is interested in doing a topological analysis of a function on a point cloud with noise or non-uniformities in density, it becomes important to take density into account, and one simple way to do this is to treat density as an additional filtration parameter.

Second, the definition of the sublevel-offset bifiltration involves a choice of filtering direction (i.e., whether we filter by sublevel or superlevel sets). While for some functions $\delta\colon X \to \mathbb{R}$, like codensity, it is natural to fix a direction, for other functions like time or partial charge, it can be unnatural to fix a filtering direction, and one would like a more symmetric construction. To this end, one can consider the *interlevel-offset trifiltration* of δ, defined as follows:

▶ **Definition 2.** *For a function $\delta : X \to \mathbb{R}$, the* interlevel-offset trifiltration *of δ, denoted $\mathcal{O}^{\updownarrow}(\delta)$, is the collection of Euclidean subspaces $(\mathcal{O}^{\updownarrow}(\delta)_{a,b,r})_{(a,b,r)\in\mathbb{R}^2\times\mathbb{R}^+}$, where $\mathcal{O}^{\updownarrow}(\delta)_{a,b,r} \coloneqq \mathcal{O}\big(\{x \in X \mid a \le \delta(x) \le b\}\big)_r$.*

Note that $\mathcal{O}^{\updownarrow}(\delta)$ is a special case of the sublevel-offset trifiltration, namely $\mathcal{O}^{\updownarrow}(\delta) = \mathcal{O}(\gamma)$, where $\gamma = (\delta, -\delta)$. We are especially interested in computing the homology of the interlevel-offset filtration in the case that δ is a time function, as this could potentially be useful in the analysis of dynamic data such as that arising in the study of collective motion of animals [1], in the study of dynamical systems, or in the sliding-window analysis of time series [50]. While $\mathcal{O}^{\updownarrow}(\delta)$ is only one of several possible choices of a multifiltration that can be built from time-varying data (e.g., see [41, 55]), it is notable for its flexibility – unlike some alternatives, the definition does not require a consistent labeling of the points at each time step, and it accommodates different numbers of points at different time steps.

Our third motivation is the following: The *degree-Rips bifiltration* has emerged in recent years as a popular choice of density-sensitive bifiltration on metric data [52, 10, 45]. The construction is appealing in part because it requires no parameter choice, whereas the sublevel-Rips filtration of a (co)density function requires a choice of bandwidth parameter. In the same way, one can define parameter-free *degree-offset* and *degree-Čech filtrations*. We are interested in the problem of computing an analogous *degree-Delaunay filtration*

which is topologically equivalent to the degree-offset bifiltration and more computable than degree-Rips or degree-Čech for low-dimensional data. In ongoing work on this problem, we noticed that the problems of computing the sublevel Delaunay(-Čech) trifiltration and the degree-Delaunay bifiltrations are very closely related. As computing the former is somewhat simpler but already rather involved, we view the results of this paper as a key step towards the development of computationally efficient degree-Delaunay bifiltrations. However, degree-Delaunay bifiltrations will not be studied in this paper.

Other Related Work. Our work and the prior work [3] solve particular instances of a general computational problem: Given a filtered point cloud $Z = (Z_p)_{p\in P}$ indexed by a poset P, one has an associated offset filtration indexed by the product poset $P \times [0, \infty)$; the problem is to define and compute a generalized Delaunay filtration of reasonable size which is weakly equivalent to this offset filtration. The case that Z is of the form $X \to X \cup Y \leftarrow Y$ was studied by Reani and Bobrowski [51], who call the resulting generalized Delaunay filtration the *coupled α-filtration*. A more general case, where instead of considering two point clouds and their union, one considers $n \geq 2$ point clouds and their union, was treated in [27, 7]. The authors call the resulting Delaunay objects *chromatic α-complexes*.

Many works have studied time-varying point cloud or metric data through the lens of persistence. To avoid the difficulties of multiparameter persistence, such work most often either considers persistence with respect to a scale parameter and ignores persistence across the time parameter [49, 1, 59, 35, 48, 22], or fixes a scale parameter and considers extended or zig-zag persistence across time [56, 43, 42]. In particular, the problem of extending Delaunay complexes to time-varying data has been studied in [38, 29] using 1-parameter (extended) persistent homology with a fixed radius parameter. It is natural to simultaneously consider persistence with respect to both scale and time, working in the multiparameter setting. This was previously explored by Kim and Mémoli [41], who introduce and study a Rips trifiltration for time-varying metric data with consistent labels across time, and by Saunders [53] and Dey and Samaga [26], who study a version of multiparameter persistence indexed over a product of a zigzag poset and a totally ordered set.

Finally, we note that a construction similar to the one used in [3] was previously used by Sheehy to construct sparse Delaunay filtrations [54].

Outline. Section 2 covers preliminaries that are needed throughout the paper. In Section 3, we define the incremental Delaunay complex $\mathcal{I}$ of a function $\gamma\colon X \to \mathbb{R}^2$, thereby completing the definition of the trifiltrations $\mathcal{D}(\gamma)$ and $\mathcal{DC}(\gamma)$. Section 4 introduces conflict pairs and presents our characterization of $\mathcal{I}$ in terms of these. In Section 5, we apply this characterization to obtain an algorithm for computing $\mathcal{I}$. We also discuss optimizations to this algorithm. In Section 6, we establish that $\mathcal{D}(\gamma)$ and $\mathcal{DC}(\gamma)$ are weakly equivalent to the sublevel offset trifiltration $\mathcal{O}(\gamma)$. Finally, in Section 7, we describe our implementation and report the results of our experiments.

2 Preliminaries

We fix a finite set $X \subset \mathbb{R}^d$. For a set $Q \subset X$, a $(d-1)$-sphere S is a *circumsphere* of Q if all points of Q lie on S, and the circumsphere S is *empty* in X if no point of $X \setminus Q$ lies inside S. We assume X is in general position, i.e., every non-empty subset Q of X of at most $d+1$ points is affinely independent, and no point of $X \setminus Q$ lies on the smallest circumsphere of Q.

The *Delaunay triangulation* of X is the simplicial complex

$$\mathrm{Del}(X) = \{\sigma \subset X \mid \sigma \text{ is non-empty and has an empty circumsphere}\}.$$

The *Voronoi cell* of a point $x \in X$ is the region

$$\mathrm{Vor}(x, X) = \{y \in \mathbb{R}^d \mid \|y - x\| \leq \|y - x'\|, \forall x' \in X\}.$$

$\mathrm{Del}(X)$ is equal to the *nerve* of the collection of Voronoi cells $\{\mathrm{Vor}(x, X) \mid x \in X\}$; see Section 6 for the definition of the nerve. For $x \in X$ and $r \geq 0$, the *Voronoi ball* $\mathcal{V}(x, X)_r$ is the region

$$\mathcal{V}(x, X)_r = \mathrm{Vor}(x, X) \cap \mathrm{Ball}_r(x),$$

where $\mathrm{Ball}_r(x) := \{y \in \mathbb{R}^| \|x - y\| \leq r\}$ is the closed ball of radius r centered at X. Letting $\mathcal{D}(X)_r$ denote the nerve of the collection of Voronoi balls $\{\mathcal{V}(x, X)_r\}_{x \in X}$, we call $\mathcal{D}(X) := (\mathcal{D}(X)_r)_{r \in \mathbb{R}^+}$ the *Delaunay filtration* of X.

We also fix a function $\gamma = (\gamma_1, \gamma_2) \colon X \to \mathbb{R}^2$. For simplicity, throughout the paper, we assume that both γ_1 and γ_2 are injective. This assumption entails no loss of generality: one can perturb γ to enforce such injectivity, carry out our trifiltration constructions for the perturbed function, and then readily recover a correct result for the unperturbed function from this. For $i = 1, 2$, let $<_i$ denote the total order on X induced by γ_i, and let $\max_i(\sigma)$ be the maximum element of σ with respect to $<_i$.

Throughout the paper, we consider the product partial order on $\mathbb{R}^2$, given by $p \leq q$ if and only if $p_1 \leq q_1$ and $p_2 \leq q_2$. We say p, q are *comparable* if $p \leq q$ or $q \leq p$; otherwise, they are *incomparable*. For $p \in \mathbb{R}^2$, let $X_p := \{x \in X \mid \gamma(x) \leq p\}$ denote the p-sublevel set of γ.

3 Delaunay trifiltrations

▶ **Definition 3.** *The* incremental Delaunay complex*, denoted $\mathcal{I}$ is the simplicial complex whose simplices are the non-empty subsets $\sigma \subset X$ with a circumsphere S of $\sigma \setminus \{\max_1(\sigma), \max_2(\sigma)\}$ satisfying:*

- *$\max_1(\sigma)$ and $\max_2(\sigma)$ are either inside or on S,*
- *every $x \in X \setminus \sigma$ such that $\gamma(x) < \gamma(\sigma)$ is either outside S or on S.*

For a simplex $\sigma \in \mathcal{I}$, we call the infimal radius of such a sphere S the incremental Delaunay radius*, and denote this as $\omega(\sigma)$.*

Note that in Section 3, we can have $\max_1(\sigma) = \max_2(\sigma)$, in which case $\gamma(\sigma) = \gamma(\max_1(\sigma))$.

▶ **Definition 4.** *For $\gamma : X \to \mathbb{R}^2$, the* sublevel Delaunay trifiltration $\mathcal{D}(\gamma)$ *and* sublevel Delaunay-Čech trifiltration $\mathcal{DC}(\gamma)$ *are defined over $\mathbb{R}^2 \times \mathbb{R}^+$ as follows:*

$$\mathcal{D}(\gamma)_{p,r} := \{\sigma \in \mathcal{I} \mid \gamma(\sigma) \leq p \text{ and } \omega(\sigma) \leq r\},$$
$$\mathcal{DC}(\gamma)_{p,r} := \{\sigma \in \mathcal{I} \mid \gamma(\sigma) \leq p \text{ and } m(\sigma) \leq r\},$$

where $\omega(\sigma)$ denotes the incremental Delaunay radius of σ (see Definition 3) and $m(\sigma)$ is the radius of the smallest enclosing ball of σ.

▶ **Proposition 5.** *For $r \geq 0$ and $p \in \mathbb{R}^2$, we have $\mathcal{D}(X_p)_r \subset \mathcal{D}(\gamma)_{p,r}$.*

Taking r sufficiently large, Proposition 5 implies that $\mathcal{I}$ contains the Delaunay triangulation $\mathrm{Del}(X_p)$ for all $p \in \mathbb{R}^2$.

4 Conflict pairs and triples

We now characterize the simplices of $\mathcal{I}$ in a way that is conducive to computation.

Let $\mathrm{Grid}(\gamma) = \mathrm{im}\,\gamma_1 \times \mathrm{im}\,\gamma_2$, and consider $p = (p_1, p_2) \in \mathrm{Grid}(\gamma)$. If $p_1 < \max(\mathrm{im}\,\gamma_1)$, then let $p{\rightarrow} = (p_1^+, p_2)$, where p_1^+ is the element of $\mathrm{im}\,\gamma_1$ immediately after p_1. We define variants of this notation, e.g., $p{\uparrow}$ and $p_{\downarrow}$, in the analogous way. For $p \in \mathrm{Grid}(\gamma)$ with $p_1 < \max(\mathrm{im}\,\gamma_1)$ and $p_2 < \max(\mathrm{im}\,\gamma_2)$, we let $p^{\nearrow} = (p{\uparrow}){\rightarrow}$.

▶ **Definition 6** (Conflict pairs and triples)**.**

(i) *For $p \in \mathrm{Grid}(\gamma)$, a* conflict pair at p *is a pair (τ, x) such that*

- **a.** *τ is a d-simplex in $\mathrm{Del}(X_p)$,*
- **b.** *either*
  - *$(p_1 < \max(\mathrm{im}\,\gamma_1)$ and $x \in X_{p{\rightarrow}})$ or*
  - *$(p_2 < \max(\mathrm{im}\,\gamma_2)$ and $x \in X_{p{\uparrow}})$,*
- **c.** *x lies inside the circumsphere of τ.*

(ii) *For distinct conflict pairs (τ, x) and (τ, y) at p, we call (τ, x, y) a* conflict triple at p.

(iii) *We call (τ, x) a* conflict pair *if it is a conflict pair at some p, and similarly for conflict triples.*

Definition 6 is illustrated in Figure 3. Note that if (τ, x, y) is a conflict triple at p, then by the injectivity of γ_1 and γ_2, either $x \in X_{p{\rightarrow}}$ and $y \in X_{p{\uparrow}}$, or else $y \in X_{p{\rightarrow}}$ and $x \in X_{p{\uparrow}}$.

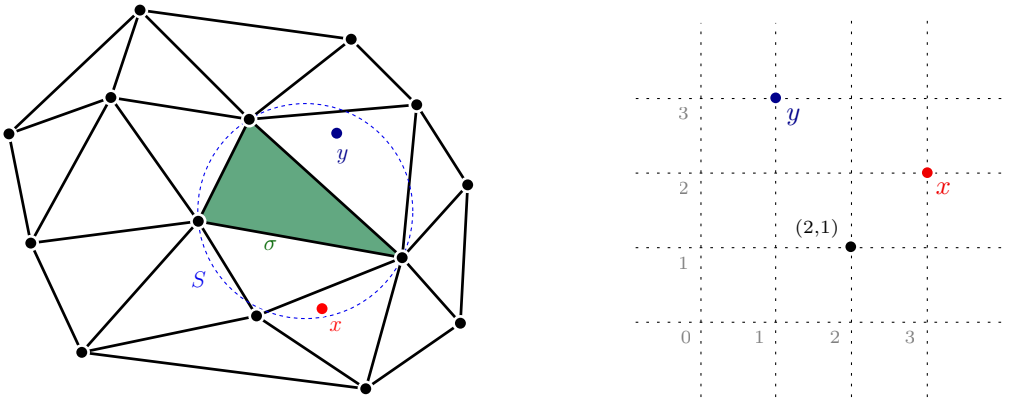


Figure 3 Let $\mathrm{Del}(X_{(2,1)})$ be as shown in black on the left, and let $\sigma \in \mathrm{Del}(X_{(2,1)})$ be the green 2-simplex with circumsphere S. Since $X_{(2,1)} = X_{(2,2)}$, we have $\mathrm{Del}(X_{(2,1)}) = \mathrm{Del}(X_{(2,2)})$, so $\sigma \in \mathrm{Del}(X_{(2,2)})$. At index $p = (2, 2)$, the point x (red) lies inside S, and belongs to $X_{p{\rightarrow}} = X_{(3,2)}$, so (σ, x) is a conflict pair at $(2, 2)$. Similarly, the point y (blue) lies inside S, and belongs to $X_{p{\uparrow}} = X_{(2,3)}$, so (σ, y) is a conflict pair at $(2, 2)$. Since (σ, x) and (σ, y) are distinct conflict pairs at $(2, 2)$, (σ, x, y) is a conflict triple at $(2, 2)$.

▶ **Lemma 7.** *If (σ, x) and (σ, y) are distinct conflict pairs, then $\gamma(x)$ and $\gamma(y)$ are incomparable.*

Proof. To arrive at a contradiction, suppose $\gamma(x) < \gamma(y)$, and (σ, y) is a conflict pair at p. Then $\gamma(x) < \gamma(y) \leq (p^{\nearrow})$. Thus, by injectivity of γ_1 and γ_2, $\gamma(x) \leq p$. Since (σ, x) is a conflict pair, we have $\sigma \notin \mathrm{Del}(X_p)$, which contradicts that (σ, y) is a conflict pair at p. ◀

Lemma 7 implies that for any d-simplex σ in $\bigcup_p \mathrm{Del}(X_p)$, the set

$$T_\sigma \coloneqq \{x \mid (\sigma, x) \text{ is a conflict pair}\}$$

is totally ordered by γ_1-coordinate. Henceforth, we assume that T_σ is given this order.

▶ **Lemma 8.** *For $x, y \in T_\sigma$, (σ, x, y) is a conflict triple if and only if x, y are ordered consecutively in T_σ.*

Proof. If x, y are ordered consecutively, then it is easily checked that (σ, x, y) is a conflict triple at p, where $p^{\nearrow} = \gamma(x) \vee \gamma(y)$. Conversely, if there exists $z \in T_\sigma$ with $x < z < y$ in the order on T_σ, then for any p with $x \in X_{p\uparrow}$ and $y \in X_{p\rightarrow}$, we have $z \in X_p$, so $\sigma \notin \mathrm{Del}(X_p)$. Hence (σ, x, y) is not a conflict triple. ◀

The next lemma follows immediately from the definition of $\mathcal{I}$.

▶ **Lemma 9.**
(i) *If (τ, x) is a conflict pair, then $\tau \cup \{x\}$ is a $(d+1)$-simplex in $\mathcal{I}$.*
(ii) *If (τ, x, y) is a conflict triple, then $\tau \cup \{x, y\}$ is a $(d+2)$-simplex in $\mathcal{I}$.*

In a way that the following proposition makes precise, every simplex is a face of a $(d+2)$-simplex or $(d+1)$-simplex coming from conflicts as above. For this, we must assume that X satisfies the Δ-*property*, i.e., the total orders $<_1$ and $<_2$ coincide in the first $d+1$ points and X is contained in the convex hull of these first $d+1$ points. This technical condition can be easily enforced in practice by simply adding $d+1$ artificial points "at infinity." The analogous condition for a single total order (or a slight variant thereof) is commonly exploited in the incremental computation of Delaunay triangulations [31, 11, 12], and is also used in the previous work on sublevel-Delaunay bifiltrations [3].

▶ **Proposition 10.** *If X has the Δ-property and $|X| \geq d+2$, then every $\tau \in \mathcal{I}(\gamma)$ is either*
- *a face of the $(d+2)$-simplex $\sigma \cup \{x, y\}$ for some conflict triple (σ, x, y), or*
- *a face of the $(d+1)$-simplex $\sigma \cup \{x\}$ for some conflict pair (σ, x) where $|T_\sigma| = 1$.*

Size. We apply the above results, together with a result about the incremental construction of Delaunay triangulations [31], to bound the size of $\mathcal{I}$:

▶ **Theorem 11.** *For constant dimension d, $\mathcal{I}$ has $O(|X|^{\lceil (d+1)/2 \rceil + 1})$ simplices.*

5 Computation

By Proposition 10, to compute $\mathcal{I}$ it suffices to compute all conflict pairs and conflict triples. In the following, we explain how to compute all conflict pairs. Once we have done so, we can easily compute all conflict triples via Lemma 8, by iterating through the set of conflict pairs T_σ for each d-simplex $\sigma \in \bigcup_{p \in \mathrm{Grid}(\gamma)} \mathrm{Del}(X_p)$.

The Bowyer-Watson algorithm. We make essential use of the *Bowyer-Watson algorithm* [15, 57], a standard incremental algorithm for computing Delaunay triangulations. Given the Delaunay triangulation $\mathrm{Del}(Y)$ of a point cloud $Y \subset \mathbb{R}^d$ and a point $y \in \mathbb{R}^d$, the Bowyer-Watson algorithm computes $\mathrm{Del}(Y \cup \{y\})$. To do so, it first identifies each d-simplex σ such that y lies inside the circumsphere of σ. Each such simplex σ is then removed from $\mathrm{Del}(Y)$, resulting in a star-convex region R centered at y. Finally, to obtain $\mathrm{Del}(Y \cup \{y\})$, the region R is retriangulated by forming the simplicial cone with base the boundary of R and apex y (See Figure 4). For σ a d-simplex as above, we call the pair (σ, y) a *BW-conflict (for Y)*.

▶ Remark 12. Note that the definition of BW-conflict is distinct from, but closely related to, that of a conflict pair given in Definition 6 (i). Namely, every conflict pair (σ, y) at p is a BW-conflict for X_p, and conversely, if $y \in X_{p\rightarrow} \cup X_{p\uparrow}$ and (σ, y) is a BW-conflict for X_p, then (σ, y) is a conflict pair.

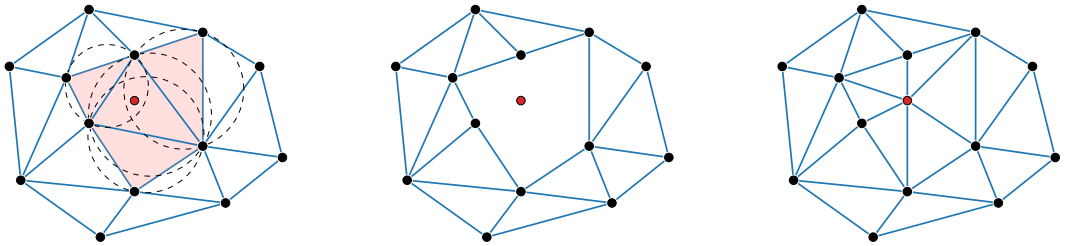


Figure 4 The Delaunay triangulation $\mathrm{Del}(Y)$ (blue), and the new point y (red). Triangles whose circumcircles contain y are shaded. We remove all these triangles, leaving an untriangulated star-convex region R centered at y. R is then retriangulated by the simplicial cone with base the boundary of R and apex y, yielding $\mathrm{Del}(Y \cup \{y\})$.

Naive Computation of Conflict Pairs. To compute all conflict pairs, a natural strategy is to traverse each horizontal and each vertical line in $\mathrm{Grid}(\gamma)$ in increasing order, iteratively applying the Bowyer-Watson algorithm. To explain this, we focus on the traversal of vertical lines, as the case of horizontal lines is symmetric. A naive version of the strategy, which already satisfies our main complexity bound (Theorem 13), is as follows: For each $a \in \operatorname{im}\gamma_1$, iterate through $\operatorname{im}\gamma_2 = \{b_1, \ldots, b_n\}$ in increasing order. For each $i < n$, let $p = (a, b_i)$. Given $\mathrm{Del}(X_p)$, if $X_{p\uparrow} \setminus X_p$ is non-empty, hence equal to $\{x\}$ for some $x \in X$, then we apply Bowyer-Watson to compute both $\mathrm{Del}(X_{p\uparrow})$ and all conflict pairs of the form (σ, x) at p.

▶ **Theorem 13.** *The above approach computes $\mathcal{I}$ in time $O(|X|^{\lceil d/2 \rceil + 2})$.*

Computing trifiltrations. Once $\mathcal{I}$ is computed, completing the computation of either $\mathcal{D}(\gamma)$ or $\mathcal{DC}(\gamma)$ amounts to computing the birth index of each simplex $\sigma \in \mathcal{I}$ in the trifiltration. Recall that the birth index of σ in $\mathcal{DC}(\gamma)$ is $(\gamma(\sigma), m_\sigma)$, where m_σ is the radius of the smallest enclosing ball of σ. Computation of the first two components of this triple is trivial. In general, smallest enclosing balls can be computed via standard algorithms with efficient implementations [58, 34, 33, 36, 32]. Assuming d is constant, computing m_σ also requires constant time. Further, recall that the birth index of σ in $\mathcal{D}(\gamma)$ is $(\gamma(\sigma), \omega_\sigma)$, where ω_σ is the incremental Delaunay radius of Definition 3. We discuss the computation of ω_σ in the full version, where we observe that ω_σ can be computed for all $\sigma \in \mathcal{I}$ in amortized constant time per simplex. From this and Theorem 13, we obtain the following:

▶ **Corollary 14.** *Both $\mathcal{D}(\gamma)$ and $\mathcal{DC}(\gamma)$ can be computed in time $O(|X|^{\lceil d/2 \rceil + 2})$.*

5.1 Optimized computation of conflict pairs

The above approach to computing conflict pairs involves considerable redundant computation across different vertical lines. We next discuss strategies for reducing this redundancy.

Whereas the unoptimized algorithm described above uses only point insertions in Delaunay triangulations, our optimizations use both insertions and deletions. Our aim in selecting these optimizations is to reduce the total number $\zeta(X)$ of insertions and deletions, which can be seen heuristically as a proxy for the total cost of the algorithm in practice. Our optimizations do not reduce $\zeta(X)$ in the worst case, and do not improve the asymptotic worst-case complexity. But in some classes of examples they yield an asymptotic reduction of $\zeta(X)$.

First, we note that for any $x \in X$, it suffices to begin the traversal of the vertical line containing $\gamma(x)$ at index $p = \gamma(x)_\downarrow$, rather than at index $(\gamma_1(x), \min_{y \in X} \gamma_2(y))$. However, to exploit this idea, one must select an algorithm to compute $\mathrm{Del}(X_{\gamma(x)_\downarrow})$. A simple approach

is to use the Bowyer-Watson algorithm with randomized insertion of points, which is known to be highly efficient in expectation [21]. However, this requires the same number of point insertions as the unoptimized algorithm, which constructs $\mathrm{Del}(X_{\gamma(x)_\downarrow})$ via point insertions in order of γ_2-value. We next introduce an alternative algorithm to compute $\mathrm{Del}(X_{\gamma(x)_\downarrow})$ which exploits the work done along the previous vertical line to reduce $\zeta(X)$ in favorable cases.

Computing $\mathbf{Del}(\boldsymbol{X_{\gamma(x)_\downarrow}})$ via deletions. Assume that we handle vertical lines in rightward order. Consider $w, x \in X$, with w immediately before x in the order $<_1$. We thus handle the vertical line containing $\gamma(w)$ immediately before the one containing $\gamma(x)$. Let W denote the final sublevel set of γ encountered as we traverse the line containing $\gamma(w)$, i.e., $W = \{y \in X \mid \gamma_1(y) \leq \gamma_1(w)\}$. Noting that $X_{\gamma(x)_\downarrow} \subset W$, let $R = W \setminus X_{\gamma(x)_\downarrow}$. We compute $\mathrm{Del}(X_{\gamma(x)_\downarrow})$ from $\mathrm{Del}(W)$ by removing the points of R one by one, using the standard algorithm of [25] to repair the Delaunay triangulation after each removal.

The number of vertex removals required to obtain $X_{\gamma(x)_\downarrow}$ from $\mathrm{Del}(W)$ is at most the number of insertions required for the subsequent traversal of the vertical line containing $\gamma(x)$, because each vertex removed must be reinserted. Thus, in all instances, this strategy increases $\zeta(X)$ by at most a factor of two, compared to the unoptimized algorithm. Further, the next example shows that this strategy can lead to an asymptotic reduction in $\zeta(X)$ on a certain family of inputs.

▶ **Example 15.** If the product partial order on $\mathbb{R}^2$ restricts to a total order on X, then $\zeta = O(|X|)$ for the optimized strategy described above, but $\zeta = O(|X|^2)$ for the unoptimized strategy. On the other hand, if the elements of X are pairwise incomparable, then $\zeta = O(|X|^2)$ for both strategies.

Reducing the number of insertions and deletions. In fact, to compute the conflict pairs first appearing on the vertical line containing $\gamma(x)$, it is not necessary to explicitly remove all points of R, but only a subset $Q \subset R$. This idea leads to a further optimization, which we call the *local algorithm*, that further reduces $\zeta(X)$. To distinguish the approaches, we call the optimized algorithm described above the *non-local algorithm*.

The local algorithm is illustrated in Figure 5. The algorithm proceeds as follows: We maintain a Delaunay triangulation T, to be modified by point removals and insertions. First, we compute $T = \mathrm{Del}(W \cup \{x\})$ from $\mathrm{Del}(W)$ via the Bowyer-Watson algorithm. We next iteratively remove from T the neighbors of x that belong to R, in arbitrary order, until x has no neighbor in R. Letting Q denote the set of removed points, after these removals, we have $T = \mathrm{Del}((W \cup \{x\}) \setminus Q)$. We then remove x from T. Finally, we insert the points of $Q \cup \{x\}$ into T in increasing order of γ_2-value, noting the BW-conflicts that arise. Each BW-conflict (σ, y) such that $x \in \sigma \cup \{y\}$ is recorded as a conflict pair.

The correctness of this approach hinges on the fact that Delaunay triangulations are local, in the sense of the following lemma. For a simplicial complex K and vertex v, the *star* of v in K, denoted $\mathrm{Star}(v, K)$, is the set of simplices of K that contain v.

▶ **Lemma 16.** *Let $Z \subset \mathbb{R}^d$ be a finite point set in general position and $z \in Z$ be a point. Let $V_z = \mathcal{N}(z, \mathrm{Del}(Z))$. Then, for any Y such that $V_z \subset Y \subset Z$, we have*

$$\mathrm{Star}(z, \mathrm{Del}(V_z)) = \mathrm{Star}(z, \mathrm{Del}(Y)).$$

▶ **Proposition 17.** *The local algorithm correctly computes all conflict pairs.*

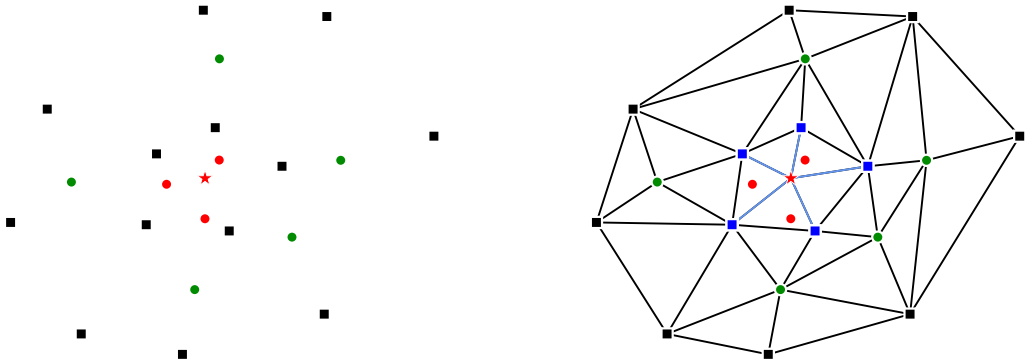


Figure 5 The left figure shows the point set W prior to inserting x (red star). Points of R are shown as green and red circles; these are the points of W with larger γ_2-value than x. Points of $X_{\gamma(x)\downarrow}$ are shown as squares; these are the points of W with smaller γ_2-value. In the non-local algorithm, all points of R are removed from the Delaunay triangulation and then re-inserted in increasing γ_2-order. In the local algorithm, points of R that are Delaunay neighbors of x (red circles) are iteratively removed until we arrive at a configuration where every Delaunay neighbor of x lies in $W \setminus R$ (blue squares, right figure). We next remove x, and then re-insert the set S of removed points (red) in increasing γ_2-order.

6 Topological equivalence of trifiltrations

Our main theoretical justification for computing Delaunay-(Čech) trifiltrations is the following:

▶ **Theorem 18.** *The trifiltrations $\mathcal{D}(\gamma)$, $\mathcal{DC}(\gamma)$, and $\mathcal{O}(\gamma)$ are all weakly equivalent.*

We begin by recalling the definition of weak equivalence; in homotopy theory, this is a standard notion of "homotopical equivalence" of diagrams of topological spaces. For a poset $\mathcal{P}$, a *$\mathcal{P}$-indexed filtration* is a collection of topological spaces $F = (F_p)_{p\in\mathcal{P}}$ such that $F_p \subset F_q$, whenever $p \leq q$. Equivalently, F is a functor from the poset category of $\mathcal{P}$ to the category of topological spaces **Top**.

▶ **Definition 19.**

(i) *Given $\mathcal{P}$-indexed filtrations F, F', a* natural transformation $\psi\colon F \to F'$ *is a collection of continuous maps $(\psi_p : F_p \to F'_p)_{p\in\mathcal{P}}$ such that the following diagram commutes for all $p \leq q$.*

$$\begin{array}{ccc} F_p & \hookrightarrow & F_q \\ \downarrow & & \downarrow \\ F'_p & \hookrightarrow & F'_q. \end{array}$$

(ii) *We call ψ a* pointwise homotopy equivalence *if each ψ_p is a homotopy equivalence.*

(iii) *We say that F and F' are* weakly equivalent*, and write $F \simeq F'$, if there exists a zigzag of pointwise homotopy equivalences connecting F and F':*

$$F \leftarrow G^1 \rightarrow G^2 \leftarrow G^3 \rightarrow \cdots \leftarrow G^k \rightarrow F'.$$

To show that $\mathcal{D}(\gamma) \simeq \mathcal{O}(\gamma)$, we characterize $\mathcal{D}(\gamma)$ as the nerve of a cover of a certain trifiltration $\mathcal{TO}$ in $\mathbb{R}^d \times \mathbb{R}^2$, which we call the *telescopic offset trifiltration*. A functorial version of the nerve theorem [6] then implies that $\mathcal{D}(\gamma) \simeq \mathcal{TO}$. In addition, a simple deformation retraction argument gives that $\mathcal{TO} \simeq \mathcal{O}(\gamma)$. Hence, $\mathcal{D}(\gamma) \simeq \mathcal{O}(\gamma)$. Similar constructions have previously been used to establish weak equivalence of 1-parameter and 2-parameter filtrations in [19, 23]. Our proof that $\mathcal{DC}(\gamma) \simeq \mathcal{O}(\gamma)$ is very similar; it uses a description of Delaunay-Čech complexes as nerves, which extends a construction of Blaser and Brun [8].

In [3], the analogous results for sublevel Delaunay(-Čech) bifiltrations were instead proven via discrete Morse theory, by extending an argument of Bauer and Edelsbrunner [5] to the 2-parameter setting. While we expect that approach to extend to our 3-parameter setting, we find our nerve-based approach to be more intuitive. In what follows, we outline the proof that $\mathcal{D}(\gamma) \simeq \mathcal{O}(\gamma)$. The proof that $\mathcal{DC}(\gamma) \simeq \mathcal{O}(\gamma)$ appears entirely in the full version.

Functorial nerve theorem for closed, semi-algebraic sets. A *cover* of a set $X \subset \mathbb{R}^d$ is a set $C = \{C(i)\}_{i \in I}$ of subsets of X whose union is X. The *nerve* of C is the simplicial complex

$$\mathrm{Nrv}(C) = \left\{ \sigma \subset I \mid \sigma \neq \emptyset,\ \bigcap_{i \in \sigma} C(i) \neq \emptyset \right\}.$$

We say C is a *good* cover if every intersection of cover elements is either empty or contractible.

These definitions extend naturally to $\mathcal{P}$-indexed filtrations, as follows: For F a $\mathcal{P}$-indexed filtration, a *cover* of F is a collection of $\mathcal{P}$-indexed filtrations $\mathcal{C} = \{\mathcal{C}(i)\}_{i \in I}$, such that for each $p \in \mathcal{P}$, $\{\mathcal{C}(i)_p\}_{i \in I}$ is a cover of F_p. We say that $\mathcal{C}$ is *good* if for each p, $\{\mathcal{C}(i)_p\}_{i \in I}$ is a good cover of F_p. We define $\mathrm{Nrv}(\mathcal{C})$, the *nerve of* $\mathcal{C}$, to be the $\mathcal{P}$-indexed filtration given by $\mathrm{Nrv}(\mathcal{C})_p = \mathrm{Nrv}(\{\mathcal{C}(i)_p\}_{i \in I})$, with structure maps the inclusions.

We will use the following functorial version of the nerve theorem for semi-algebraic sets; see also [6] for a thorough treatment of other variants of the nerve theorem.

▶ **Theorem 20.** *Let $\mathcal{C} = \{\mathcal{C}(i)\}_{i \in I}$ be a finite, good cover of a $\mathcal{P}$-indexed filtration F such that for each $p \in \mathcal{P}$ and $i \in I$, $\mathcal{C}(i)_p$ is closed and semi-algebraic. Then $F \simeq \mathrm{Nrv}(\mathcal{C})$.*

Proof. This follows immediately from [6, Theorem 5.9], using the fact that an inclusion of closed semi-algebraic sets satisfies the homotopy extension property [24, Theorem 4]. ◀

▶ Remark 21. Theorem 20 is proven in [6, Theorem 4.10] under the additional assumption that each $\mathcal{C}(i)_p$ is compact, via a somewhat different argument.

Telescopic offset trifiltration. For $(p, r) \in \mathbb{R}^2 \times \mathbb{R}^+$, the *telescopic offset* $\mathcal{TO}_{p,r} \subset \mathbb{R}^d \times \mathbb{R}^2$ is given by

$$\mathcal{TO}_{p,r} \coloneqq \bigcup_{q \leq p} \mathcal{O}(\gamma)_{q,r} \times \{q\}. \tag{1}$$

Equivalently,

$$\mathcal{TO}_{p,r} = \bigcup_{x \in X} \mathrm{Ball}_r(x) \times [\gamma(x), p],$$

where $[\gamma(x), p] = \{q \in \mathbb{R}^2 \mid \gamma(x) \leq q \leq p\}$. Varying p and r, we obtain the *telescopic offset trifiltration* $\mathcal{TO} \colon \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbf{Top}$. Let $\psi \colon \mathcal{TO} \to \mathcal{O}(\gamma)$ be the natural transformation induced by the projection of $\mathbb{R}^d \times \mathbb{R}^2$ onto the first d coordinates.

▶ **Lemma 22.** *The natural transformation ψ is a pointwise homotopy equivalence.*

A cover by telescopic Voronoi balls. We next define our cover of $\mathcal{TO}$. For $x \in X$, we define the *telescopic Voronoi cell*

$$\mathrm{Tel}(x) \coloneqq \mathrm{cl}\Big(\bigcup_{\gamma(x) \le q} \mathrm{Vor}(x, X_q) \times \{q\} \Big),$$

where cl denotes the closure. For $(p, r) \in \mathbb{R}^2 \times \mathbb{R}^+$, we define the *telescopic Voronoi ball*

$$\begin{aligned} \mathcal{T}(x)_{p,r} &\coloneqq \mathrm{cl}\Big(\bigcup_{\gamma(x) \le q \le p} \mathcal{V}(x, X_q)_r \times \{q\} \Big) \\ &= \mathrm{Tel}(x) \cap (\mathrm{Ball}_r(x) \times \{y \in \mathbb{R}^2 \mid y \le p\}). \end{aligned}$$

See Figure 6 for an illustration in the case $d = 1$. Allowing p and r to vary, the spaces $\mathcal{T}(x)_{p,r}$ assemble into a trifiltration $\mathcal{T}(x) \colon \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbf{Top}$. We let $\mathcal{T} = \{\mathcal{T}(x)\}_{x \in X}$.

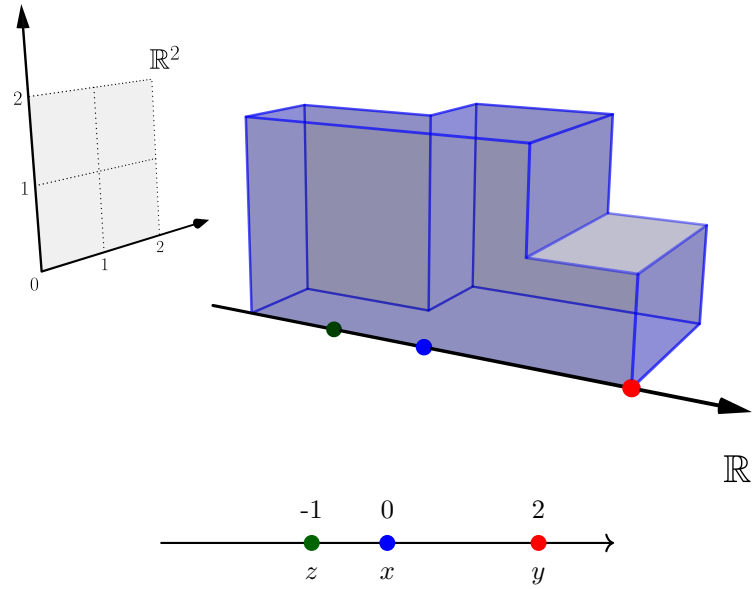


Figure 6 The point set X consists of three points x (blue), y (red), and z (green) in $\mathbb{R}$, whose function values are $\gamma(x) = (0, 0)$, $\gamma(y) = (0, 1)$, and $\gamma(z) = (1, 0)$. The telescopic Voronoi ball $\mathcal{T}(x)_{p,r}$ of radius $r = 2$ at $p = (2, 2)$ for the point x (blue) is shown as the blue shaded polyhedron.

▶ **Lemma 23.** *The set $\mathcal{T}$ is a good cover of $\mathcal{TO}$.*

In what follows, we identify simplices of $\mathrm{Nrv}(\mathcal{T})$ with their corresponding subsets of X.

▶ **Lemma 24.** *The trifiltrations $\mathrm{Nrv}(\mathcal{T})$ and $\mathcal{D}(\gamma)$ are equal.*

▶ **Proposition 25.** *We have $\mathcal{D}(\gamma) \simeq \mathcal{O}(\gamma)$.*

7 Implementation

We have written a C++ program to compute the *Delaunay-Čech trifiltration* $\mathcal{DC}(\gamma)$ for a function $\gamma : X \to \mathbb{R}^2$, where $X \subset \mathbb{R}^d$. The code is available on Bitbucket[1]. Our code builds upon a prior implementation of the algorithm from [3] for computing the Delaunay-Čech filtration of an $\mathbb{R}$-valued function. Our program accepts a text file as input, where each line represents the coordinate of a point $x \in \mathbb{R}^d$ along with its function value $\gamma(x)$, and outputs a chain complex representation of $\mathcal{DC}(\gamma)$ in the `scc2020` format, as described in [40]. Both the local and non-local algorithms are implemented. Our implementation computes the incremental Delaunay complex essentially as described in Section 5. However, there are minor differences, which arise because some simplifications that have been incorporated into this paper have not yet been implemented in the code. As in [3], simplices are stored in a simplex tree [13, 47], and minimum enclosing balls are computed in CGAL [32].

Experiments. The test suite we use, obtained from [39], consists of point clouds obtained by sampling 1-spheres (S^1) and unit squares ($[0,1]^2$) in $\mathbb{R}^2$, and 2-spheres (S^2), tori ($S^1 \times S^1$) and unit cubes ($[0,1]^3$) in $\mathbb{R}^3$, with 5% noise drawn from a uniform distribution and perturbation to ensure general position. We consider point clouds of 500, 1000, 2000, 4000, 8000, 16000 points, with four point clouds per type and size.

For each point cloud X we computed the interlevel Delaunay-Čech trifiltration of four different functions $\delta : X \to \mathbb{R}$, using the local algorithm. The four functions are:

- *codensity*, $\delta(x) = -\sum_{y \neq x} \exp\left(-\frac{\|x-y\|^2}{\sigma^2}\right)$, with σ chosen as the 0.1^{th} percentile of the non-zero distances between points in X,
- L_1-*coeccentricity*, $\delta(x) = -\sum_{y \in X} \frac{\|x-y\|}{|X|}$,
- *height*, $\delta(x) = x_d$, where x_d is the last coordinate of x,
- *random*, where for each $x \in X$, $\delta(x)$ is chosen uniformly at random from $[0,1]$.

All experiments were performed on a computer with an Intel Core i7-5960X CPU @3.00GHz and 64GB of memory, running Ubuntu 20.04.6 LTS. The code was compiled with `g++ 9.4.0`.

The full results of the experiments are available on Zenodo[2]. A representative subset of the results is given in Table 1; this data is plotted in Figure 7. We observe that across all examples, the size of the incremental Delaunay complex and the memory usage grow nearly linearly as a function of the input size. In contrast, the runtime grows nearly quadratically. Thus, our experiments indicate that our approach is memory efficient, and therefore that substantially larger computations should be feasible.

Table 2 compares the performance of the local and non-local algorithms on three types of examples, with $|X|$ up to 4000. While the local algorithm offers no improvement over the non-local algorithm in the first example type (interlevel trifiltrations), for the two other example types, the local algorithm is always faster by a factor of between 3 and 14, and the factor increases with the size of the data set.

[1] `https://bitbucket.org/mkerber/function_delaunay` (archived here)
[2] `https://doi.org/10.5281/zenodo.19227801`

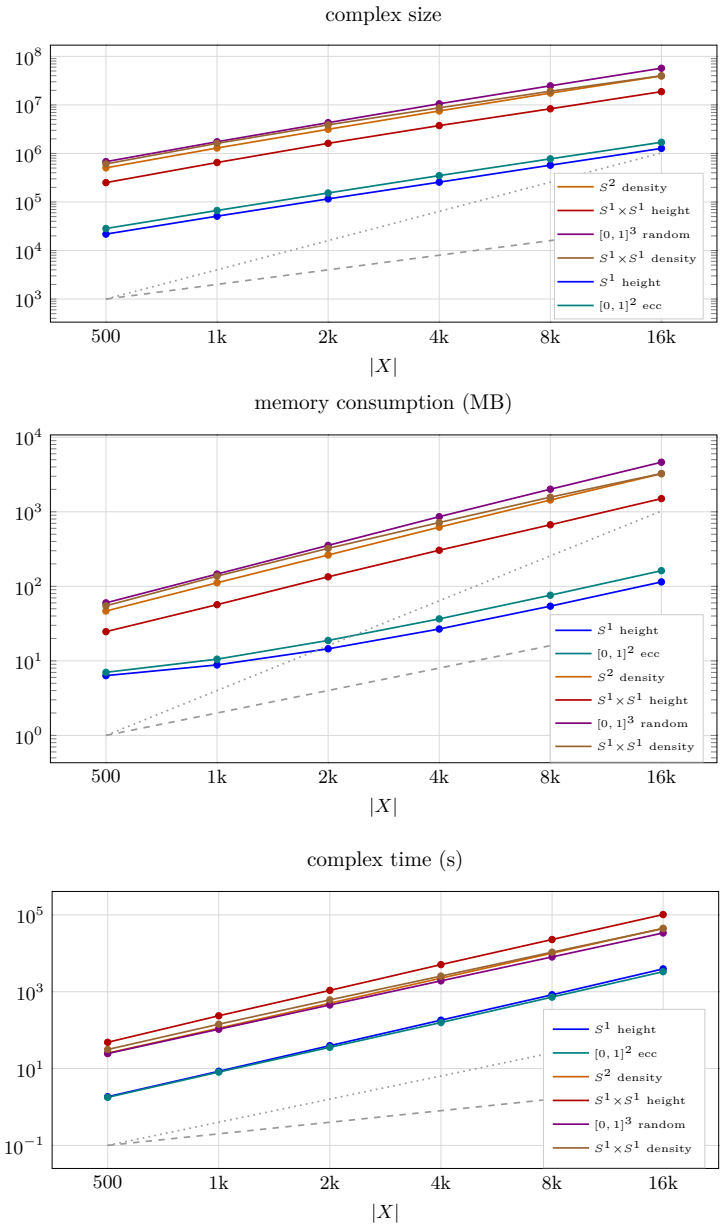


Figure 7 Log-log plots of complex size (top), memory consumption in MB (middle), and time in seconds (bottom) as a function of input size $|X|$, for the computations reported in Table 1. For reference, in each plot, the graphs of a line $y = c_1 x$ and quadratic $y = c_2 x^2$ are shown as dashed and dotted lines, respectively, for some choices of c_1 and c_2 that vary among the three plots. The slope of these lines is determined by the corresponding exponent. In the top plot, for each type of example, the sizes (y-values) are well fit by a translate of the line representing $y = c_1 x$, reflecting that the sizes grow nearly linearly. In the middle plot, the same is very nearly true, indicating that memory consumption also grows nearly linearly. In the bottom plot, for each type of example, the runtimes (y-values) are well fit by a translate of the line representing $y = c_2 x^2$, reflecting that the runtimes grow nearly quadratically.

Table 1 Complex size, running time (in seconds), and the memory consumption (in MB) for computing the interlevel Delaunay-Čech trifiltration via the local algorithm. We also report the size of the Delaunay complex and the time required to compute it.

(X, δ)	$\lvert X\rvert$	Complex $\mathcal{I}(\gamma)$			Chain complex		Delaunay	
		Size	Memory	Time	Memory	Time	Size	Time
S^1, height	500	21763	6.34	1.85	11.96	1.90	2971	0.00
	1000	50967	8.82	8.49	21.78	8.61	5975	0.00
	2000	115775	14.51	39.42	43.88	39.71	11971	0.00
	4000	256043	26.76	181.79	90.72	182.48	23959	0.01
	8000	572891	54.20	831.17	196.03	832.96	47967	0.03
	16000	1270049	114.66	3924.57	427.51	3928.97	95953	0.08
$[0, 1]^2$, eccentricity	500	28415	7.02	1.78	14.28	1.85	2973	0.00
	1000	66775	10.54	8.10	27.24	8.26	5971	0.00
	2000	153219	18.80	35.71	56.42	36.09	11971	0.00
	4000	349689	36.57	157.55	125.85	158.56	23969	0.01
	8000	773433	75.99	723.86	270.45	726.45	47963	0.03
	16000	1698421	162.12	3319.30	593.21	3325.91	95951	0.09
S^2, density	500	505301	46.64	25.16	173.62	26.89	12961	0.03
	1000	1295335	111.53	112.14	440.04	116.94	26439	0.09
	2000	3145589	263.06	501.26	1069.18	514.21	54327	0.03
	4000	7514870	621.77	2249.95	2517.43	2283.56	109987	0.07
	8000	17501199	1439.11	10064.50	5783.57	10147.90	222339	0.15
	16000	39523383	3239.97	44737.40	13028.31	44927.60	450133	0.42
$S^1 \times S^1$, height	500	250593	24.66	48.22	87.41	48.91	13765	0.01
	1000	650151	56.75	236.34	225.28	238.47	27003	0.04
	2000	1615669	134.14	1082.43	548.63	1088.31	52717	0.02
	4000	3751973	304.95	5075.66	1247.05	5090.92	105249	0.05
	8000	8316989	669.48	22835.40	2729.30	22870.80	211885	0.12
	16000	18764567	1502.64	101974	6150.01	102059	426531	0.30
$[0, 1]^3$, random	500	677493	60.00	24.56	231.95	26.91	13701	0.04
	1000	1741733	146.14	105.52	592.96	112.56	27609	0.13
	2000	4309321	354.44	451.97	1435.99	470.73	56299	0.03
	4000	10550531	859.78	1916.20	3466.08	1963.40	113975	0.07
	8000	24740775	2008.65	8030.43	8326.79	8146	227809	0.18
	16000	57038595	4623.33	33816.30	19116.62	34114	458947	0.59
$S^1 \times S^1$, density	500	607060	54.73	31.32	208.71	33.41	13765	0.04
	1000	1614337	136.96	142.33	549.31	148.57	27003	0.12
	2000	3885310	322.87	613.11	1304.49	629.68	52717	0.03
	4000	8711827	717.71	2548.13	2879.86	2587.38	105249	0.07
	8000	19140397	1571.01	10684.30	6305.07	10772.40	211885	0.16
	16000	39868943	3265.80	43863.60	13121.97	44053	426531	0.43

Table 2 Comparison of the non-local and local algorithms across several datasets. In the interlevel case (S^1, $\gamma_1 = \text{height}$, $\gamma_2 = -\text{height}$), the local algorithm removes all points of R, but incurs a slight overhead from searching for vertices to remove, resulting in marginally worse running times than the non-local algorithm. In the other cases, the local algorithm performs substantially better.

(X, γ)	$\lvert X\rvert$	Time (s)		Ratio
		non-local algorithm	local algorithm	
S^1 $\gamma_1 = \text{height}$ $\gamma_2 = -\text{height}$	500	1.76	1.88	0.93
	1000	8.10	8.61	0.94
	2000	37.64	39.72	0.95
	4000	174.58	181.88	0.96
$[0,1]^3$ $\gamma_1 = \text{random}$ $\gamma_2 = \text{random}$	500	13.83	4.13	3.35
	1000	56.01	11.36	4.93
	2000	226.25	29.61	7.64
	4000	962.79	70.97	13.57
$S^1 \times S^1$ $\gamma_1 = \text{density}$ $\gamma_2 = \text{random}$	500	18.43	5.45	3.38
	1000	82.00	16.79	4.88
	2000	340.38	44.98	7.57
	4000	1416.12	124.28	11.39

References

1 Henry Adams, Veronica Ciocanel, Chad M Topaz, and Lori Ziegelmeier. Topological data analysis of collective motion. *SIAM News*, 53(01), 2020.

2 Ángel Javier Alonso. A Sparse Multicover Bifiltration of Linear Size. In *41st International Symposium on Computational Geometry (SoCG 2025)*, 2025. doi:10.4230/LIPIcs.SoCG.2025.6.

3 Ángel Javier Alonso, Michael Kerber, Tung Lam, and Michael Lesnick. Delaunay Bifiltrations of Functions on Point Clouds. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 4872–4891, Philadelphia, PA, January 2024. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611977912.

4 Ángel Javier Alonso, Michael Kerber, and Siddharth Pritam. Filtration-domination in bifiltered graphs. In *2023 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 27–38. SIAM, 2023. doi:10.1137/1.9781611977561.CH3.

5 Ulrich Bauer and Herbert Edelsbrunner. The morse theory of čech and delaunay complexes. *Transactions of the American Mathematical Society*, 369(5):3741–3762, 2017.

6 Ulrich Bauer, Michael Kerber, Fabian Roll, and Alexander Rolle. A unified view on the functorial nerve theorem and its variations. *Expositiones Mathematicae*, 41(4):125503, 2023. doi:10.1016/j.exmath.2023.04.005.

7 Ranita Biswas, Sebastiano Cultrera di Montesano, Ondřej Draganov, Herbert Edelsbrunner, and Morteza Saghafian. On the Size of Chromatic Delaunay Mosaics, 2022. doi:10.48550/arXiv.2212.03121.

8 Nello Blaser and Morten Brun. Relative persistent homology. *Discrete & Computational Geometry*, 68(4):949–963, 2022. doi:10.1007/S00454-022-00421-9.

9 Nello Blaser, Morten Brun, Odin Hoff Gardaa, and Lars M. Salbu. Core bifiltration. *Journal of Applied and Computational Topology*, 9(4):30, November 2025. doi:10.1007/s41468-025-00226-8.

10 Andrew J Blumberg and Michael Lesnick. Stability of 2-parameter persistent homology. *Foundations of Computational Mathematics*, 24(2):385–427, 2024. doi:10.1007/S10208-022-09576-6.

11 Jean-Daniel Boissonnat, Olivier Devillers, and Samuel Hornus. Incremental construction of the delaunay triangulation and the delaunay graph in medium dimension. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 208–216, 2009. doi:10.1145/1542362.1542403.

12 Jean-Daniel Boissonnat, Olivier Devillers, Monique Teillaud, and Mariette Yvinec. Triangulations in cgal. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 11–18, 2000.

13 Jean-Daniel Boissonnat and Clément Maria. The simplex tree: An efficient data structure for general simplicial complexes. *Algorithmica*, 70:406–427, 2014. doi:10.1007/S00453-014-9887-3.

14 Magnus Botnan and Michael Lesnick. An introduction to multiparameter persistence. In *Representations of Algebras and Related Structures*, pages 77–150. EMS Press, 2023.

15 A. Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166, February 1981. doi:10.1093/comjnl/24.2.162.

16 Zixuan Cang, Lin Mu, and Guo-Wei Wei. Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening. *PLoS computational biology*, 14(1):e1005929, 2018. doi:10.1371/JOURNAL.PCBI.1005929.

17 Zixuan Cang and Guo-Wei Wei. Persistent cohomology for data with multicomponent heterogeneous information. *SIAM Journal on Mathematics of Data Science*, 2(2):396–418, 2020. doi:10.1137/19M1272226.

18 Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009. doi:10.1007/S00454-009-9176-0.

19 Nicholas J Cavanna, Mahmoodreza Jahanseir, and Donald R Sheehy. A geometric perspective on sparse filtrations. *arXiv preprint arXiv:1506.03797*, 2015. arXiv:1506.03797.

20 Frédéric Chazal, Leonidas J Guibas, Steve Y Oudot, and Primoz Skraba. Scalar field analysis over point cloud data. *Discrete & Computational Geometry*, 46(4):743–775, 2011. doi:10.1007/S00454-011-9360-X.

21 Kenneth L Clarkson and Peter W Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, pages 387–421, 1989.

22 David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. Vines and vineyards by updating persistence in linear time. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 119–126, 2006. doi:10.1145/1137856.1137877.

23 René Corbet, Michael Kerber, Michael Lesnick, and Georg Osang. Computing the multicover bifiltration. *Discrete & Computational Geometry*, 70(2):376–405, 2023. doi:10.1007/S00454-022-00476-8.

24 Hans Delfs and Manfred Knebusch. Separation, retractions and homotopy extension in semialgebraic spaces. *Pacific Journal of Mathematics*, 114(1):47–71, 1984.

25 Olivier Devillers. On deletion in delaunay triangulations. In *Proceedings of the fifteenth annual symposium on Computational geometry*, SoCG99, pages 181–188. ACM, June 1999. doi:10.1145/304893.304969.

26 Tamal K. Dey and Shreyas N. Samaga. Quasi zigzag persistence: A topological framework for analyzing time-varying data, 2026. doi:10.48550/arXiv.2502.16049.

27 Sebastiano Cultrera di Montesano, Ondřej Draganov, Herbert Edelsbrunner, and Morteza Saghafian. Chromatic Alpha Complexes, 2024. arXiv:2212.03128.

28 Herbert Edelsbrunner and John L Harer. *Computational topology: an introduction*. American Mathematical Society, 2022.

29 Herbert Edelsbrunner, Carl-Philipp Heisenberg, Michael Kerber, and Gabriel Krens. The medusa of spatial sorting: topological construction. *arXiv preprint arXiv:1207.6474*, 2012. arXiv:1207.6474.

30 Herbert Edelsbrunner and Ernst P Mücke. Three-dimensional alpha shapes. *ACM Transactions On Graphics (TOG)*, 13(1):43–72, 1994. doi:10.1145/174462.156635.

31 Herbert Edelsbrunner and Nimish R. Shah. Incremental topological flipping works for regular triangulations. *Algorithmica*, 15(3):223–241, 1996. doi:10.1007/BF01975867.

32 Kaspar Fischer, Bernd Gärtner, Thomas Herrmann, Michael Hoffmann, and Sven Schönherr. Bounding volumes. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6.1 edition, 2024. URL: https://doc.cgal.org/5.6.1/Manual/packages.html#PkgBoundingVolumes.

33 Kaspar Fischer and Bernd Gärtner. The smallest enclosing ball of balls: Combinatorial structure and algorithms. *Int. J. Comput. Geom. Appl.*, 14:341–378, 2004. doi:10.1142/S0218195904001500.

34 Kaspar Fischer, Bernd Gärtner, and Martin Kutz. Fast Smallest-Enclosing-Ball Computation in High Dimensions. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003*, volume 2832, pages 630–641. Springer Berlin Heidelberg, 2003. doi:10.1007/978-3-540-39658-1_57.

35 İsmail Güzel, Elizabeth Munch, and Firas A Khasawneh. Detecting bifurcations in dynamical systems with crocker plots. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(9), 2022.

36 Bernd Gärtner. Fast and Robust Smallest Enclosing Balls. In Jaroslav Nešetřil, editor, *Algorithms - ESA' 99*, volume 1643, pages 325–338. Springer Berlin Heidelberg, 1999. doi:10.1007/3-540-48481-7_29.

37 Michael Kerber. function_delaunay. https://bitbucket.org/mkerber/function_delaunay, 2025.

38 Michael Kerber and Herbert Edelsbrunner. 3d kinetic alpha complexes and their implementation. In *2013 Proceedings of the Fifteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 70–77. SIAM, 2013. doi:10.1137/1.9781611972931.6.

39 Michael Kerber and Tung Lam. Benchmark data sets of minimal presentations of 2-parameter persistence modules, 2025. doi:10.3217/rxedk-qyq77.

40 Michael Kerber and Michael Lesnick. scc2020: A file format for sparse chain complexes in tda. https://bitbucket.org/mkerber/chain_complex_format/, 2021. Accessed: 2024-12-04.

41 Woojin Kim and Facundo Mémoli. Spatiotemporal persistent homology for dynamic metric spaces. *Discrete & Computational Geometry*, 66:831–875, 2021. doi:10.1007/S00454-019-00168-W.

42 Woojin Kim and Facundo Mémoli. Extracting persistent clusters in dynamic data via möbius inversion. *Discrete & Computational Geometry*, 71(4):1276–1342, 2024. doi:10.1007/S00454-023-00590-1.

43 Woojin Kim, Facundo Memoli, and Zane Smith. Analysis of dynamic graphs and dynamic metric spaces via zigzag persistence. In *Topological Data Analysis: The Abel Symposium 2018*, pages 371–389. Springer, 2020.

44 Michael Lesnick and Kenneth McCabe. Nerve models of subdivision bifiltrations. *arXiv preprint arXiv:2406.07679*, 2024.

45 Michael Lesnick and Matthew Wright. Interactive visualization of 2-d persistence modules. *arXiv preprint arXiv:1512.00180*, 2015. arXiv:1512.00180.

46 Baihan Lin. Topological data analysis in time series: Temporal filtration and application to single-cell genomics. *Algorithms*, 15(10):371, October 2022. doi:10.3390/a15100371.

47 Clément Maria. Filtered complexes. In *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.10.0 edition, 2024. URL: https://gudhi.inria.fr/doc/3.10.0/group__simplex__tree.html.

48 Dmitriy Morozov. *Homological illusions of persistence and stability*. PhD thesis, Duke University, 2008.

49 Elizabeth Munch. *Applications of persistent homology to time varying systems*. PhD thesis, Duke University, 2013.

50 Jose A Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of computational mathematics*, 15:799–838, 2015. doi:10.1007/S10208-014-9206-Z.

51 Yohai Reani and Omer Bobrowski. A coupled alpha complex. *Journal of Computational Geometry*, 14(1):221–256, 2023. doi:10.20382/jocg.v14i1a9.

52 Alexander Rolle and Luis Scoccola. Stable and consistent density-based clustering via multiparameter persistence. *Journal of Machine Learning Research*, 25(258):1–74, 2024.

53 Emily Lauren Saunders. *Persistent Homology of Zig-Zag Families of Filtrations*. PhD thesis, Columbia University, 2025. doi:10.7916/XHZ0-VG67.

54 Donald R. Sheehy. A Sparse Delaunay Filtration. In Kevin Buchin and Éric Colin de Verdière, editors, *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 58:1–58:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2021.58.

55 Kristian Strommen, Matthew Chantry, Joshua Dorrington, and Nina Otter. A topological perspective on weather regimes. *Climate Dynamics*, 60(5):1415–1445, 2023.

56 Sarah Tymochko, Elizabeth Munch, and Firas A Khasawneh. Using zigzag persistent homology to detect hopf bifurcations in dynamical systems. *Algorithms*, 13(11):278, 2020. doi:10.3390/A13110278.

57 D. F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24(2):167–172, February 1981. doi:10.1093/comjnl/24.2.167.

58 Emo Welzl. Smallest enclosing disks (balls and ellipsoids). In Hermann Maurer, editor, *New Results and New Trends in Computer Science*, volume 555, pages 359–370, Berlin/Heidelberg, 1991. Springer-Verlag. doi:10.1007/BFb0038202.

59 Lu Xian, Henry Adams, Chad M Topaz, and Lori Ziegelmeier. Capturing dynamics of time-varying data via topology. *Foundations of Data Science*, 4(1):1–36, 2022.

Estimating the Persistent Homology of $\mathbb{R}^n$-Valued Functions Using Function-Geometric Multifiltrations

Ethan André
École Normale Supérieure, Paris, France

Jingyi Li
Inria Saclay and École Polytechnique, Palaiseau, France

David Loiseaux
Inria Saclay, Palaiseau, France

Steve Oudot
Inria Saclay and École Polytechnique, Palaiseau, France

Abstract

Given an unknown $\mathbb{R}^n$-valued function f on a metric space X, can we approximate the persistent homology of f from a finite sampling of X with known pairwise distances and function values? This question has been answered in the case $n = 1$, assuming f is Lipschitz continuous and X is a sufficiently regular geodesic metric space, and using filtered geometric complexes with fixed scale parameter for the approximation. In this paper we answer the question for arbitrary n, under similar assumptions and using function-geometric multifiltrations. Our analysis offers a different view on these multifiltrations by focusing on their approximation properties rather than on their stability properties. We also leverage the multiparameter setting to provide insight into the influence of the scale parameter, whose choice is central to this type of approach. From a practical standpoint, we show that our approximation results are robust to input noise, and that function-geometric multifiltrations have good statistical convergence properties. We also provide an algorithm to compute our estimators, and we use its implementation to conduct extensive experiments, on both synthetic and real biological data, in order to validate our theoretical results.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases Topological data analysis, multi-parameter persistent homology, function-Rips multifiltration

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.6

Related Version *Full Version*: https://arxiv.org/abs/2412.04162 [1]

Supplementary Material *Software*: https://doi.org/10.5281/zenodo.19135090

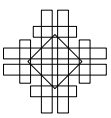
Funding *David Loiseaux*: Inria Action Exploratoire PreMediT (Precision Medicine using Topology).

1 Introduction

Let X be a metric space and $f\colon X \to \mathbb{R}^n$ a function, both unknown. Suppose we are given a finite point cloud P sampled from X, such that the pairwise distances between the points of P are known, as well as the function values at the points of P. Given this input, can we approximate the persistent homology $H_*(f)$ of f (i.e., the persistence module induced in homology from the multifiltration of X by the sublevel sets of f) with provable guarantees? This question was addressed in [10] in the case $n = 1$. The authors proposed an estimator based on a nested pair of Rips complexes $\mathcal{R}^\delta(P) \subseteq \mathcal{R}^{2\delta}(P)$, where δ is a user-defined parameter, which they filtered by the values of f at the vertices using lower-star filtrations. The estimator was then the image of the morphism of 1-parameter persistence modules



© Ethan André, Jingyi Li, David Loiseaux, and Steve Oudot;
licensed under Creative Commons License CC-BY 4.0



induced in homology by the inclusion $\mathcal{R}^\delta(P) \hookrightarrow \mathcal{R}^{2\delta}(P)$. We denote by $H_*(\mathcal{R}^{\delta\to 2\delta}(\hat{f}|_P))$ this estimator. Assuming both the domain X and the function f are sufficiently regular (typically, X is a compact geodesic metric space with positive convexity radius ϱ_X, and f is c-Lipschitz continuous), and P is an ε-sample of X in the geodesic distance for some small enough value of ε (specifically, $\varepsilon < \varrho_X/4$), they proved that the estimator is $2c\delta$-interleaved with $H_*(f)$ for any choice of parameter δ within the range $[2\varepsilon, \varrho_X/2)$. Thus, in cases where ε is known or can be estimated reliably, one can get an $O(2c\varepsilon)$-interleaving with the target $H_*(f)$, hence an $O(2c\varepsilon)$-matching with its barcode, by the algebraic stability theorem [2, 8, 9]. Under a reasonable sampling model, ε goes to zero as the number of sample points goes to infinity, hence so does the approximation error, which means that the proposed estimator is consistent. This approach has since been applied in various contexts, notably in clustering where the algorithm ToMATo [11] is an instance of the method in homology degree 0. However, two fundamental questions were left open in [10, 11]:

Q1 how to estimate ε, or to choose parameter δ directly, when ε is unknown;

Q2 how to generalize the approach to arbitrary n, i.e., to vector-valued functions f.

Q2 is precisely the question we asked at the beginning. Q1 is related to the general problem of scale parameter estimation, which has received a lot of attention in statistics, including in the context of topological data analysis (TDA) – see e.g. [4, 6, 23].

Contributions

In this paper we address both Q1 and Q2 using a unified persistence-based approach.

For Q2 we study the extension of the estimator $H_*(\mathcal{R}^{\delta\to 2\delta}(\hat{f}|_P))$ from [10] to vector-valued functions $f\colon X \to \mathbb{R}^n$, where the codomain $\mathbb{R}^n$ is equipped with the product order, and we generalize its approximation guarantee to this setting. For the sake of generality we assume f to be ω-continuous for some modulus of continuity ω, that is: $\|f(x) - f(y)\|_\infty \leq \omega(d_X(x,y))$ for any $x, y \in X$, where $\omega\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a non-decreasing subadditive function such that $\omega(\delta)$ goes to 0 as δ does. This includes Lipschitz or Hölder continuous functions as special cases. Under this assumption, our first main result (Theorem 3) states that the estimator $H_*(\mathcal{R}^{\delta\to 2\delta}(\hat{f}|_P))$ is $\omega(2\delta)$-interleaved with its target $H_*(f)$ for any choice of δ within the range $[2\varepsilon, \varrho_X/2)$, under the same regularity condition on X and ε-sampling condition on P as before. Our proof takes inspiration from the one in [10] but uses a new, purely diagrammatic formulation that clarifies it and makes it hold for any $n \geq 1$.

For Q1 we propose two complementary approaches. The first one infers a relevant choice of δ under an appropriate statistical model. This model assumes the points P are sampled i.i.d. according to some unkown probability distribution μ that is supposed to be (a, b)-standard. When parameters a, b are known, we show that there is an explicit optimal choice δ_k for parameter δ, as a function of the sample size k, that makes the sequence of approximations $\left(H_*(\mathcal{R}^{\delta_k\to 2\delta_k}(\hat{f}|_P))\right)_{k\in\mathbb{N}}$ a quasi-minimax estimator of $H_*(f)$. When a, b are unknown, as would generally be the case in practice, we show that δ_k can be effectively estimated by some $\hat{\delta}_k$ under our statistical model, so that the sequence $\left(H_*(\mathcal{R}^{\hat{\delta}_k\to 2\hat{\delta}_k}(\hat{f}|_P))\right)_{k\in\mathbb{N}}$ is again a quasi-minimax estimator of $H_*(f)$. These statements are summarized in Theorem 5.

Our second approach to addressing Q1 is meant for cases where our statistical model does not hold and ε cannot be estimated reliably. We follow the general philosophy underpinning persistent homology by letting δ vary over $\mathbb{R}_{\geq 0}$ and treating it as an extra parameter for persistence. This means that our estimator now becomes the $(n+1)$-parameter persistence module $H_*(\mathcal{R}^{\bullet\to 2\bullet}(\hat{f}|_P))$ indexed over $\mathbb{R}_{\geq 0} \times \mathbb{R}^n$, where the notation $\bullet$ indicates that the Rips parameter is also a filtration parameter. In order to enable comparisons, the target $H_*(f)$

must also be extended to $\mathbb{R}_{\geq 0} \times \mathbb{R}^n$, which we do by taking its left Kan extension $\mathrm{Lan}_\iota H_*(\mathscr{f})$ along the poset embedding $\iota : \mathbb{R}^n \xrightarrow{\cong} \{0\} \times \mathbb{R}^n \hookrightarrow \mathbb{R}_{\geq 0} \times \mathbb{R}^n$. The extension $\mathrm{Lan}_\iota H_*(\mathscr{f})$ merely contains copies of $H_*(\mathscr{f})$ in each vertical hyperplane $\{\delta\} \times \mathbb{R}^n$, connected by horizontal identities, as illustrated in Figure 1. Our third main result (Theorem 6) states that the n-dimensional interleavings happening between $H_*(\mathscr{f})$ and $H_*(\mathcal{R}^{\delta \to 2\delta}(\mathscr{f}|_P))$ inside each vertical hyperplane $\{\delta\} \times \mathbb{R}^n$ for $\delta \in [2\varepsilon, \varrho_X/2)$ commute with the horizontal morphisms in $H_*(\mathcal{R}^{\bullet \to 2\bullet}(\mathscr{f}|_P))$ and in $\mathrm{Lan}_\iota H_*(\mathscr{f})$, so that, within any vertical slab $[2\varepsilon, \delta_0] \times \mathbb{R}^n$ with $\delta_0 < \varrho_X/2$, they all together form a *vertical* $\omega(2\delta_0)$*-interleaving* (i.e., an $\omega(2\delta_0)$-interleaving along the direction $(0, 1, \cdots, 1)^\top$ in $\mathbb{R}_{\geq 0} \times \mathbb{R}^n$) between $H_*(\mathcal{R}^{\bullet \to 2\bullet}(\mathscr{f}|_P))$ and $\mathrm{Lan}_\iota H_*(\mathscr{f})$.

Since vertical interleavings are stronger than ordinary interleavings (Remark 1), Theorem 6 not only provides approximation guarantees for any interleaving-stable invariant of $\mathrm{Lan}_\iota H_*(\mathscr{f})$ by the corresponding invariant of $H_*(\mathcal{R}^{\bullet \to 2\bullet}(\mathscr{f}|_P))$, but it also offers refined statements for invariants that behave in a particular way under vertical interleavings, such as for instance multigraded Betti numbers as proven in the full version [1] and illustrated in Figure 2.

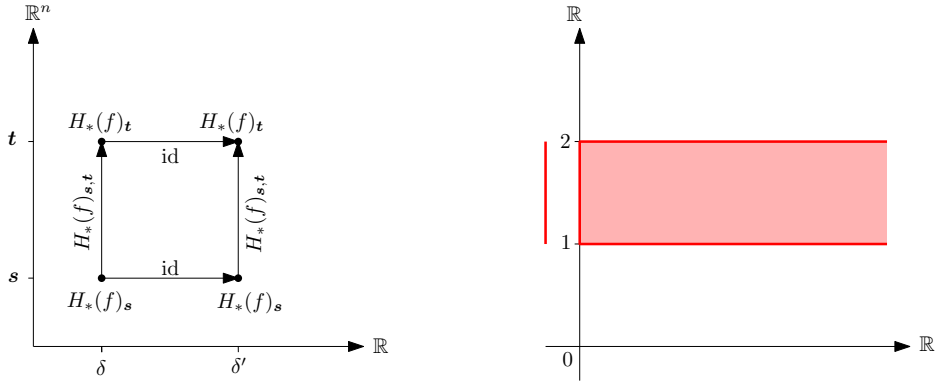


Figure 1 Left: an illustration of the left Kan extension $\mathrm{Lan}_\iota H_*(\mathscr{f}) : \mathbb{R}_{\geq 0} \times \mathbb{R}^n \to \mathbf{vec}$, with identity maps horizontally and the structural morphisms of $H_*(\mathscr{f})$ vertically. **Right**: (case $n = 1$) the left Kan extension of the interval module $\Bbbk^{[1,2]}$ is the interval module $\Bbbk^{\mathbb{R}_{\geq 0} \times [1,2]}$ (in red).

In practice, function values may be corrupted with measurement noise, meanwhile geodesic distances may have to be approximated from the input data. Our framework can account for these imprecisions by suitably adapting the parameters of the estimators and their approximation bounds (see the full version [1, Section 3.3]).

Still for practical purposes, we discuss the computation of our estimators, in particular we provide an algorithm for computing a free presentation of $H_*(\mathcal{R}^{\bullet \to 2\bullet}(\mathscr{f}|_P))$, and thus also of $H_*(\mathcal{R}^{\delta \to 2\delta}(\mathscr{f}|_P))$ for any fixed $\delta \geq 0$, in time that is comparable to that of computing a free presentation of the persistent homology of a single multifiltration. From a free presentation, a variety of invariants, including multigraded Betti numbers, can be derived using existing software. We have implemented our algorithm and used it in our experiments.

Structure of the paper

We provide background material in Section 2. Our problem statement, estimators, and main results are detailed in Section 3. In Section 4 we discuss algorithmic aspects. Finally, in Section 5 we present experimental results that illustrate our theoretical guarantees and we investigate additional properties of our estimators. Further algorithmic details, as well as the proofs of our main results, can be found in the full version [1].

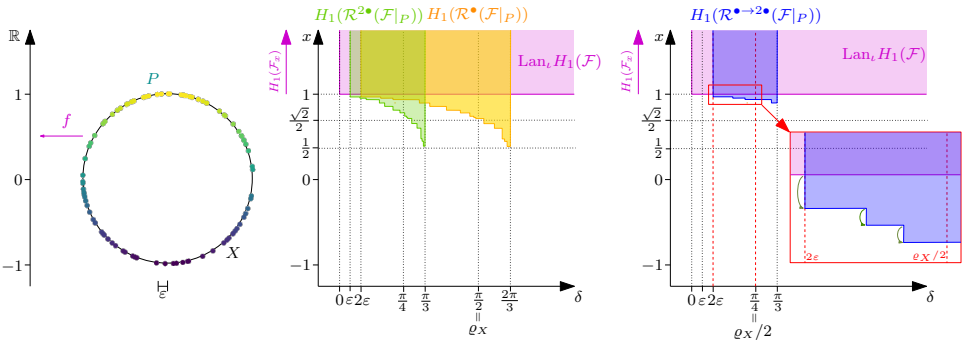


Figure 2 Left: the vertical height function on a sampled unit circle X in the plane. Distances on the circle are given by arclength. **Center**: $H_1(\mathscr{f})$ has a single interval summand (in magenta), starting at height 1, which extends to the free module $\mathrm{Lan}_\iota H_1(\mathscr{f})$ generated at $(0,1)$ in $\mathbb{R}_{\geq 0} \times \mathbb{R}$. The modules $H_1(\mathcal{R}^{\bullet}(\mathscr{f}|_P))$ (in yellow) and $H_1(\mathcal{R}^{2\bullet}(\mathscr{f}|_P))$ (in green) are interval modules in this simple scenario. **Right**: the estimator $H_1(\mathcal{R}^{\bullet\to 2\bullet}(\mathscr{f}|_P))$ (in blue), which is also an interval module, approximates the target $\mathrm{Lan}_\iota H_1(\mathscr{f})$ in the vertical interleaving distance within any slab $[2\varepsilon, \delta_0] \times \mathbb{R}$ with $\delta_0 < \varrho_X/2$, as per Theorem 6. In turn, the vertical interleaving between the two modules implies a vertical matching between their multigraded Betti numbers within the slab (illustrated by green arrows in the close-up view), as per Corollary 3.8 in the full version [1].

2 Background

We assume some familiarity with basic category theory, algebraic topology, measure theory, and topological data analysis. Let **Top** denote the category of topological spaces, and **vec** (resp. **Vec**) the category of finite-dimensional (resp. all) vector spaces over a fixed field $\Bbbk$. We also fix an arbitrary degree in homology, denoted by $*$, and we write $H_*(-)$ for singular homology groups in degree $*$ with coefficients in $\Bbbk$.

Filtrations and persistence modules

We see $\mathbb{R}^n$ as the product of n copies of the real line, equipped with the product order noted $\leq$. Thus, two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ satisfy $\boldsymbol{x} \leq \boldsymbol{y}$ whenever $\boldsymbol{x}_i \leq \boldsymbol{y}_i$ for all $1 \leq i \leq n$. We denote by $(-\infty, \boldsymbol{x}]$ the downset $\{\boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{y} \leq \boldsymbol{x}\}$, and by $[\boldsymbol{x}, +\infty)$ the upset $\{\boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{y} \geq \boldsymbol{x}\}$. An *$n$-parameter filtration* is a functor $\mathcal{F} : \mathbb{R}^n \to \mathbf{Top}$ whose constituent maps are inclusions, which means that we have a topological space $\mathcal{F}_{\boldsymbol{x}}$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and an inclusion $\mathcal{F}_{\boldsymbol{x},\boldsymbol{y}} \colon \mathcal{F}_{\boldsymbol{x}} \hookrightarrow \mathcal{F}_{\boldsymbol{y}}$ for all $\boldsymbol{x} \leq \boldsymbol{y} \in \mathbb{R}^n$. An *$n$-parameter persistence module* is a functor $M : \mathbb{R}^n \to \mathbf{Vec}$, which means that we have a $\Bbbk$-vector space $M_{\boldsymbol{x}}$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and a $\Bbbk$-linear map $M_{\boldsymbol{x},\boldsymbol{y}} \colon M_{\boldsymbol{x}} \to M_{\boldsymbol{y}}$ for all $\boldsymbol{x} \leq \boldsymbol{y} \in \mathbb{R}^n$, such that $M_{\boldsymbol{x},\boldsymbol{x}} = \mathrm{Id}_{M_{\boldsymbol{x}}}$ and $M_{\boldsymbol{x},\boldsymbol{z}} = M_{\boldsymbol{y},\boldsymbol{z}} \circ M_{\boldsymbol{x},\boldsymbol{y}}$ for all $\boldsymbol{x} \leq \boldsymbol{y} \leq \boldsymbol{z} \in \mathbb{R}^n$. A *morphism* of persistence modules $M \to N$ is a natural transformation between functors, i.e., a family of linear maps $\varphi_{\boldsymbol{x}} \colon M_{\boldsymbol{x}} \to N_{\boldsymbol{x}}$ such that $N_{\boldsymbol{x},\boldsymbol{y}} \circ \varphi_{\boldsymbol{x}} = \varphi_{\boldsymbol{y}} \circ M_{\boldsymbol{x},\boldsymbol{y}}$ for all $\boldsymbol{x} \leq \boldsymbol{y} \in \mathbb{R}^n$.

Geodesic metric spaces

Throughout the paper, unless otherwise specified, (X, d_X) is a compact geodesic metric space. This means that, for all $x, y \in X$, there is a shortest continuous path from x to y in X, of length equal to $d_X(x, y) < \infty$. In particular, the space is path-connected. Let P be a finite set of points of X. Given $\varepsilon > 0$, we say P is a *geodesic ε-sample* of X if $d_{\mathrm{H}}(P, X) < \varepsilon$, where d_{H}

denotes the Hausdorff distance in (X, d_X). Define $B_X(x, r)$ as the open geodesic ball centered at $x \in X$ with radius r, given by $B_X(x, r) = \{x' \in X \mid d_X(x, x') < r\}$. A ball $B_X(x, r)$ is said to be *convex* if, for any pair of points y and y' in $B_X(x, r)$, the shortest path in X that connects y to y' is unique and included in $B_X(x, r)$. Let $\varrho(x) := \inf\{r > 0 \mid B(x, r) \text{ is not convex}\}$, and let $\varrho_X := \inf\{\varrho(x) \mid x \in X\} \geq 0$. This quantity, called the *convexity radius* of X, plays an important role because intersections of convex sets are convex and therefore contractible, so the Nerve Lemma [13, Corollary 4G.3] applies to unions of balls of radii less than ϱ_X.

Functional and geometric filtrations

Given a geodesic metric space (X, d_X) and a function $\ell\colon X \to \mathbb{R}^n$, the *sublevel filtration* $\mathcal{F} : \mathbb{R}^n \to \mathbf{Top}$ of ℓ is defined by $\mathcal{F}_{\boldsymbol{x}} := \ell^{-1}((-\infty, \boldsymbol{x}])$ for all $\boldsymbol{x} \in \mathbb{R}^n$. The *persistent homology* of $\mathcal{F}$ (and, by extension, of ℓ) is the persistence module $H_*(\mathcal{F})$ (also written $H_*(\ell)$) defined by $H_*(\mathcal{F})_{\boldsymbol{x}} := H_*(\mathcal{F}_{\boldsymbol{x}})$ and $H_*(\mathcal{F})_{\boldsymbol{x},\boldsymbol{y}} := H_*(\mathcal{F}_{\boldsymbol{x},\boldsymbol{y}})$ for all $\boldsymbol{x} \leq \boldsymbol{y} \in \mathbb{R}^n$.

Given a finite set P of points in X, and a real parameter $\delta \geq 0$, we call *δ-offset* of P the union of open geodesic balls $\mathcal{O}^\delta(P) := \bigcup_{p\in P} B_X(p, \delta)$. We call *$\delta$-Čech Complex* of P the nerve of the collection of balls $\{B_X(p, \delta) \mid p \in P\}$, defined as the abstract simplicial complex $\mathcal{C}^\delta(P) := \{\emptyset \neq \sigma \subseteq P \mid \bigcap_{p\in\sigma} B_X(p, \delta) \neq \emptyset\}$. For $\delta < \varrho_X$, the balls $B_X(p, \delta)$ and their intersections are either empty or convex, so the Nerve Lemma [13, Corollary 4G.3] ensures that the homology groups $H_*(\mathcal{O}^\delta(P))$ and $H_*(\mathcal{C}^\delta(P))$ are isomorphic. We call *δ-Rips complex* of P the abstract simplicial complex $\mathcal{R}^\delta(P) := \{\emptyset \neq \sigma \subseteq P \mid \max_{p,q\in\sigma} d_X(p, q) < \delta\}$. Čech and Rips complexes of P are related as follows: $\mathcal{C}^\delta(P) \subseteq \mathcal{R}^{2\delta}(P) \subseteq \mathcal{C}^{2\delta}(P)$ for all $\delta \geq 0$.

Interleavings

Let $\boldsymbol{v} \in \mathbb{R}^n_{\geq 0}$ be a fixed vector, and let $\varepsilon \geq 0$. Given any functors $F, G : \mathbb{R}^n \to \mathcal{C}$ to some category $\mathcal{C}$, we say F and G are *$\varepsilon\boldsymbol{v}$-interleaved* if there are two natural transformations $f : F \to G[\varepsilon\boldsymbol{v}]$ and $g : G \to F[\varepsilon\boldsymbol{v}]$ such that $g[\varepsilon\boldsymbol{v}] \circ f = \varphi_F^{2\varepsilon\boldsymbol{v}}$ and $f[\varepsilon\boldsymbol{v}] \circ g = \varphi_G^{2\varepsilon\boldsymbol{v}}$. Here, $-[\varepsilon\boldsymbol{v}]$ denotes the $\varepsilon\boldsymbol{v}$-shift functor, defined on objects by $F[\varepsilon\boldsymbol{v}]_{\boldsymbol{x}} := F_{\boldsymbol{x}+\varepsilon\boldsymbol{v}}$ and on morphisms by $F[\varepsilon\boldsymbol{v}]_{\boldsymbol{x},\boldsymbol{y}} := F_{\boldsymbol{x}+\varepsilon\boldsymbol{v},\boldsymbol{y}+\varepsilon\boldsymbol{v}}$ for all $\boldsymbol{x} \leq \boldsymbol{y} \in \mathbb{R}^n$. Meanwhile, $\varphi_F^{2\varepsilon\boldsymbol{v}}\colon F \to F[2\varepsilon\boldsymbol{v}]$ and $\varphi_G^{2\varepsilon\boldsymbol{v}}\colon G \to G[2\varepsilon\boldsymbol{v}]$ denote the natural morphisms from F and G to their respective $2\varepsilon\boldsymbol{v}$-shifts, defined by $(\varphi_F^{2\varepsilon\boldsymbol{v}})_{\boldsymbol{x}} := F_{\boldsymbol{x},\boldsymbol{x}+2\varepsilon\boldsymbol{v}}$ and $(\varphi_G^{2\varepsilon\boldsymbol{v}})_{\boldsymbol{x}} := G_{\boldsymbol{x},\boldsymbol{x}+2\varepsilon\boldsymbol{v}}$ for all $\boldsymbol{x} \in \mathbb{R}^n$. The *$\boldsymbol{v}$-interleaving distance* is defined as follows:

$$d_{\mathrm{i}}^{\boldsymbol{v}}(F, G) = \inf\{\varepsilon \geq 0 \mid \text{ there exists an } \varepsilon\boldsymbol{v}\text{-interleaving between } F \text{ and } G\} \in [0, +\infty].$$

In the following, we consider two specific types of interleavings:

1. **Ordinary interleavings**: Let $\boldsymbol{v} = \mathbf{1} := (1, \cdots, 1)^T$. Then, $\varepsilon\mathbf{1}$-interleavings are ordinary ε-interleavings from the TDA literature, and $d_{\mathrm{i}}^{\mathbf{1}}$ is called the *ordinary interleaving distance*.
2. **Vertical interleavings**: Let $\boldsymbol{v} = \mathbf{1}_0 := (0, 1, \cdots, 1)^T$. Then, $\varepsilon\mathbf{1}_0$-interleavings are called *vertical ε-interleavings*, and $d_{\mathrm{i}}^{\mathbf{1}_0}$ is called the *vertical interleaving distance*.

▶ Remark 1. Vertical interleaving implies ordinary interleaving. Indeed, the vertical interleaving maps can be composed with the horizontal structure morphisms of the modules to form an ordinary interleaving. The amplitude ε of the interleaving does not change in the process. Therefore, vertical interleaving is a stronger condition than ordinary interleaving.

3 Problem statement and main results

Let (X, d_X) be a compact geodesic metric space with positive convexity radius ϱ_X, and let $\ell : X \to \mathbb{R}^n$ be a continuous function. Here, $\mathbb{R}^n$ is equipped with the ∞-norm. Both X and ℓ are unknown. By the Heine–Cantor Theorem, ℓ admits a modulus of continuity

$\omega\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Neither ω nor ϱ_X need to be known, as they only play a role in our bounds and not in our constructions. The function $\boldsymbol{f}$ is not assumed to be pointwise finite-dimensional (i.e., $\dim H_*(\boldsymbol{f})_{\boldsymbol{x}}$ finite for every $\boldsymbol{x} \in \mathbb{R}^n$), however our assumptions imply a form of tameness for the persistence module $H_*(\boldsymbol{f})$ (see Corollary 4).

Our input is a finite point cloud $P \subseteq X$ such that $\boldsymbol{f}|_P$, the restriction of $\boldsymbol{f}$ to P, is known, as well as the pairwise geodesic distances $d_X(p,q)$ between the points $p, q \in P$. The problem we address is that of estimating $H_*(\boldsymbol{f})$, the persistent homology of the sublevel filtration of $\boldsymbol{f}$, from our input. For this we assume that P is a geodesic ε-sample of X, for some possibly unknown ε that we assume to be small enough compared to the convexity radius ϱ_X.

To build our estimator, we first construct the *function-Rips multifiltration* $\mathcal{R}^\bullet(\boldsymbol{f}|_P) := \left\{\mathcal{R}^\delta\left(\mathcal{F}_{\boldsymbol{x}} \cap P\right)\right\}_{\delta\in\mathbb{R}_{\geq 0},\, \boldsymbol{x}\in\mathbb{R}^n}$ from the restriction of $\boldsymbol{f}$ to P. Then, following [10], our estimator is a smoothed version of $H_*(\mathcal{R}^\bullet(\boldsymbol{f}|_P))$, denoted by $H_*(\mathcal{R}^{\bullet\to 2\bullet}(\boldsymbol{f}|_P))$ and defined formally as the image of the morphism of persistence modules induced in homology by the inclusion of the filtration $\mathcal{R}^\bullet(\boldsymbol{f}|_P)$ into its horizontal rescaling by a factor of 2, that is:

$$H_*(\mathcal{R}^{\bullet\to 2\bullet}(\boldsymbol{f}|_P)) := \operatorname{Im}\ H_*\left(\mathcal{R}^\bullet(\boldsymbol{f}|_P) \hookrightarrow \mathcal{R}^{2\bullet}(\boldsymbol{f}|_P)\right),$$

where $\mathcal{R}^{2\bullet}(\boldsymbol{f}|_P) := \left\{\mathcal{R}^{2\delta}(\mathcal{F}_{\boldsymbol{x}} \cap P)\right\}_{\delta\in\mathbb{R}_{\geq 0},\, \boldsymbol{x}\in\mathbb{R}^n}$.

▶ Remark 2. It is known that a single Rips complex of a finite sampling can recover the homological type of a sufficiently regular underlying continuous space [14, 16]. Yet, in full generality it takes a pair of Rips complexes with parameters within a factor of at least 2 of each other to do so [7]. Similarly, it was recently shown that $H_*(\mathcal{R}^\bullet(\boldsymbol{f}|_P))$ itself can approximate $H_*(\boldsymbol{f})$ provided X has curvature bounded above locally [22]. Yet, it is unknown whether the same holds in our more general setting, which is why we use $H_*(\mathcal{R}^{\bullet\to 2\bullet}(\boldsymbol{f}|_P))$.

In the course of our analysis of the behavior of $H_*(\mathcal{R}^{\bullet\to 2\bullet}(\boldsymbol{f}|_P))$, we will consider intermediate constructions, namely the persistent homologies of:

- the *function-offset filtration* $\mathcal{O}^\bullet(\boldsymbol{f}|_P) := \left\{\mathcal{O}^\delta(\mathcal{F}_{\boldsymbol{x}} \cap P)\right\}_{\delta\in\mathbb{R}_{\geq 0},\, \boldsymbol{x}\in\mathbb{R}^n}$;
- the *function-Čech filtration* $\mathcal{C}^\bullet(\boldsymbol{f}|_P) := \left\{\mathcal{C}^\delta(\mathcal{F}_{\boldsymbol{x}} \cap P)\right\}_{\delta\in\mathbb{R}_{\geq 0},\, \boldsymbol{x}\in\mathbb{R}^n}$.

These can also serve as alternative estimators when they can be built, for instance – in the case of $\mathcal{C}^\bullet(\boldsymbol{f}|_P)$ – when we can test whether finitely many open geodesic balls intersect.

We use our estimators in two types of scenarios: (1) when we have an estimate of the value ε for which P is an ε-sample of X, and (2) when we do not have such an estimate.

3.1 Estimating $H_*(\boldsymbol{f})$ with known ε

Suppose we know ε or some reasonable estimate. Then we can approximate our target $H_*(\boldsymbol{f})$ by restricting our estimators to a vertical hyperplane $\{\delta\} \times \mathbb{R}^n \subseteq \mathbb{R}_{\geq 0} \times \mathbb{R}^n$ for a suitable choice of parameter δ. We denote these restrictions respectively by $H_*(\mathcal{O}^\delta(\boldsymbol{f}|_P))$, $H_*(\mathcal{C}^\delta(\boldsymbol{f}|_P))$ and $H_*(\mathcal{R}^{\delta\to 2\delta}(\boldsymbol{f}|_P))$.

▶ **Theorem 3.** *Let X be a compact geodesic space, let $\boldsymbol{f} : X \to \mathbb{R}^n$ be an ω-continuous function for some modulus of continuity $\omega : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, and let P be a finite geodesic ε-sample of X. Then:*

(i) *for $\forall \delta \geq \varepsilon$, the modules $H_*(\boldsymbol{f})$ and $H_*(\mathcal{O}^\delta(\boldsymbol{f}|_P))$ are ordinarily $\omega(\delta)$-interleaved;*

(ii) *for $\forall \delta \in [\varepsilon, \varrho_X)$, the modules $H_*(\boldsymbol{f})$ and $H_*(\mathcal{C}^\delta(\boldsymbol{f}|_P))$ are ordinarily $\omega(\delta)$-interleaved;*

(iii) *for $\forall \delta \in [2\varepsilon, \frac{\varrho_X}{2})$, modules $H_*(\boldsymbol{f})$ and $H_*(\mathcal{R}^{\delta\to 2\delta}(\boldsymbol{f}|_P))$ are ordinarily $\omega(2\delta)$-interleaved.*

The proof of this result is given in the full version [1, Section 4]. The conclusion still holds under noisy input function values or geodesic distances, as detailed in [1, Section 3.3].

Theorem 3 can be read in two different ways. First, when the modulus of continuity ω is known, we obtain explicit bounds on the interleaving distance between our estimators and the target $H_*(f)$. Second, when we only know that f is continuous, the Heine–Cantor Theorem implies that f admits some modulus of continuity ω, and even without knowning that ω, we are guaranteed by Theorem 3 that our estimators converge to $H_*(f)$ as ε goes to zero under suitable choices of parameter δ.

When ε is known exactly, the best choice for δ is the lower bound of its admissible interval, i.e., $\delta = \varepsilon$ in Items i and ii and $\delta = 2\varepsilon$ in Item iii. When ε is only known approximately, the admissible interval provides some leeway for the choice of δ.

An immediate consequence of Theorem 3 is that $d_{\mathrm{i}}^{\mathbf{1}}$-stable invariants computed on our estimators approximate the corresponding invariants defined on the target $H_*(f)$. For instance, the approximation guarantee obtained for the multigraded Betti numbers of $H_*(\mathcal{R}^{\delta\to 2\delta}(f|_P))$ is given in the full version [1, Corollary 3.3].

Another consequence of Theorem 3 (i) is that, under our assumptions, the persistence module $H_*(f)$ satisfies the following form of tameness, as proven in the full version [1].

▶ **Corollary 4.** *Let X, ω, f be as in Theorem 3. Then the module $H_*(f)$ is tame in the sense that the rank $\mathrm{rk}(H_*(f)_{\boldsymbol{x},\boldsymbol{y}})$ is finite for all $\boldsymbol{x} < \boldsymbol{y} \in \mathbb{R}^n$ with $\boldsymbol{x}_i < \boldsymbol{y}_i$ for each $i \in \{1, \cdots, n\}$.*

Suppose now the points of P are i.i.d. samples drawn from some unknown probability measure μ supported on X. In this setting, standard statistical techniques can be used to estimate ε with high probability, then Theorem 3 can be applied to get statistical estimates of $H_*(f)$ or of any $d_{\mathrm{i}}^{\mathbf{1}}$-stable invariant thereof. We summarize our results in Theorem 5. The relevant statistical background and definitions are provided in the full version [1, Section 2], together with the analysis of the statistical estimators [1, Section 6].

▶ **Theorem 5.** *Let X, ω and f be as in Theorem 3. Assume that the convexity radius satisfies $\varrho_X > 0$, that the modulus of continuity ω has a controlled vanishing rate $\delta = O(\omega(\delta))$ as $\delta \to 0$, and that the points $p_1, \ldots, p_k$ of P are i.i.d. samples drawn from some unknown (a,b)-standard probability measure μ whose support is the entire space X.*

- **(i)** *Suppose a, b are known. Let $\delta_k := 4\left(\frac{2\log k}{ak}\right)^{1/b}$. Then, the sequences of approximations $\left(H_*(\mathcal{O}^{\delta_k}(f|_P))\right)_{k\in\mathbb{N}}$, $\left(H_*(\mathcal{C}^{\delta_k}(f|_P))\right)_{k\in\mathbb{N}}$ and $\left(H_*(\mathcal{R}^{\delta_k\to 2\delta_k}(f|_P))\right)_{k\in\mathbb{N}}$ of $H_*(f)$ are ω-quasi-minimax consistent estimators of $H_*(f)$ in the following sense: their expected interleaving distances to $H_*(f)$ are at most a constant times $\omega\left(8\left(\frac{2\log k}{ak}\right)^{1/b}\right)$; meanwhile, in the worst case over n, X, f as above, for any other estimator of $H_*(f)$, the supremum over all (a,b)-standard measures μ of the expected interleaving distance between the estimator and its target $H_*(f)$ is at least a constant times $\omega\left(\frac{1}{2}\left(\frac{1}{ak}\right)^{1/b}\right)$.*
- **(ii)** *Suppose a, b are unknown, and assume further that X is a compact smooth manifold and that the measure μ decomposes as $\mu = \mu_1 + \mu_2$, where $\mu_2(X) > 0$ and μ_2 is absolutely continuous with respect to the uniform measure on X, with a positive density on X. Then, there is an explicit sequence of random positive numbers $(\hat{\delta}_k)_{k\in\mathbb{N}}$ such that the induced plug-in estimators $\left(H_*(\mathcal{O}^{\hat{\delta}_k}(f|_P))\right)_{k\in\mathbb{N}}$, $\left(H_*(\mathcal{C}^{\hat{\delta}_k}(f|_P))\right)_{k\in\mathbb{N}}$, and $\left(H_*(\mathcal{R}^{\hat{\delta}_k\to 2\hat{\delta}_k}(f|_P))\right)_{k\in\mathbb{N}}$ are ω-quasi-minimax consistent estimators of $H_*(f)$.*

3.2 Estimating $H_*(f)$ with unknown ε

When ε is unknown and cannot be effectively estimated – for instance when the assumptions in Theorem 5 are not satisfied, we adopt the traditional approach in persistence theory which is to not fix the scale parameter δ but rather to consider it as an extra filtration

parameter. This means using the full $(n+1)$-parameter modules $H_*(\mathcal{O}^\bullet(f|_P))$, $H_*(\mathcal{C}^\bullet(f|_P))$ and $H_*(\mathcal{R}^{\bullet\to 2\bullet}(f|_P))$ as estimators. In order to state approximation results, we extend our target $H_*(f)$ to a persistence module over $\mathbb{R}_{\geq 0}\times\mathbb{R}^n$ by taking its left Kan extension $\mathrm{Lan}_\iota H_*(f)$ along the poset embedding $\iota:\mathbb{R}^n\hookrightarrow\mathbb{R}_{\geq 0}\times\mathbb{R}^n$ given by $\boldsymbol{x}\mapsto(0,\boldsymbol{x})$ for all $\boldsymbol{x}\in\mathbb{R}^n$. Since the category $\mathbb{R}^n$ is small and the category **Vec** is cocomplete, $\mathrm{Lan}_\iota H_*(f)$ is well-defined as a functor $\mathbb{R}_{\geq 0}\times\mathbb{R}^n\to\mathbf{Vec}$, and given pointwise by the colimit formula [19]:

$$\forall(\delta,\boldsymbol{x})\in\mathbb{R}_{\geq 0}\times\mathbb{R}^n,\quad \mathrm{Lan}_\iota H_*(f)_{(\delta,\boldsymbol{x})}=\varinjlim_{\substack{\boldsymbol{y}\in\mathbb{R}^n\\ \iota(\boldsymbol{y})\leq(\delta,\boldsymbol{x})}} H_*(f)_{\boldsymbol{y}}\cong H_*(f)_{\boldsymbol{x}}. \tag{1}$$

Moreover, $\mathrm{Lan}_\iota H_*(f)$ has structural morphisms that are identity maps horizontally (i.e., along the first coordinate axis) and the structural morphisms of $H_*(f)$ vertically (i.e., along any direction orthogonal to the first coordinate axis), as illustrated in Figure 1.

▶ **Theorem 6.** *Let X, ω, f, P and ε as in Theorem 3. Then:*

- **(i)** *for $\forall\delta_0\geq\varepsilon$, within the slab $[\varepsilon,\delta_0]\times\mathbb{R}^n$, the restricted modules $\mathrm{Lan}_\iota H_*(f)|_{[\varepsilon,\delta_0]\times\mathbb{R}^n}$ and $H_*(\mathcal{O}^\bullet(f|_P))|_{[\varepsilon,\delta_0]\times\mathbb{R}^n}$ are vertically $\omega(\delta_0)$-interleaved;*
- **(ii)** *for $\forall\delta_0\in[\varepsilon,\varrho_X)$, within the slab $[\varepsilon,\delta_0]\times\mathbb{R}^n$, the restricted modules $\mathrm{Lan}_\iota H_*(f)|_{[\varepsilon,\delta_0]\times\mathbb{R}^n}$ and $H_*(\mathcal{C}^\bullet(f|_P))|_{[\varepsilon,\delta_0]\times\mathbb{R}^n}$ are vertically $\omega(\delta_0)$-interleaved;*
- **(iii)** *for $\forall\delta_0\in[2\varepsilon,\varrho_X/2)$, within the slab $[2\varepsilon,\delta_0]\times\mathbb{R}^n$, the restricted modules $\mathrm{Lan}_\iota H_*(f)|_{[2\varepsilon,\delta_0]\times\mathbb{R}^n}$ and $H_*(\mathcal{R}^{\bullet\to 2\bullet}(f|_P))|_{[2\varepsilon,\delta_0]\times\mathbb{R}^n}$ are vertically $\omega(2\delta_0)$-interleaved.*

The proof is given in the full version [1, Section 5]. Similar to Theorem 3, the conclusion still holds under noisy input function values or geodesic distances – see [1, Section 3.3].

This approximation result encapsulates the previous one in the sense that restricting the estimators to the vertical hyperplane $\{\delta\}\times\mathbb{R}^n$ for $\delta=\delta_0$ recovers Theorem 3. However, the result says something deeper, namely: that the interleavings in the vertical hyperplanes $\{\delta\}\times\mathbb{R}^n$, for δ ranging over $[\varepsilon,\delta_0]$ (resp. $[2\varepsilon,\delta_0]$), commute with the horizontal structural morphisms of the modules, so that they all together form a vertical interleaving in the slab $[\varepsilon,\delta_0]\times\mathbb{R}^n$ (resp. $[2\varepsilon,\delta_0]\times\mathbb{R}^n$). Intuitively, this means that prominent features in $H_*(f)$ are not just present in $H_*(\mathcal{R}^{\delta\to 2\delta}(f|_P))$ at individual scales δ in the range $[2\varepsilon,\delta_0]$, but that they persist across large ranges of scales in $H_*(\mathcal{R}^{\bullet\to 2\bullet}(f|_P))$. In particular, they are not ephemeral in $\mathbb{R}_{\geq 0}\times\mathbb{R}^n$ and can be captured by stable invariants.

In terms of approximation accuracy, the result says that, the more one looks to the left (i.e., toward small positive values of δ_0) in the parameter space $\mathbb{R}_{\geq 0}\times\mathbb{R}^n$, the more precisely the estimators approximate the target, until some point where δ_0 becomes too small and the approximation breaks down. On the opposite side, the more one looks to the right (i.e., toward large values of δ_0), the more the estimators drift away from the target, until again the approximation eventually breaks down if the estimator is $H_*(\mathcal{C}^\bullet(f|_P))$ or $H_*(\mathcal{R}^{\bullet\to 2\bullet}(f|_P))$. See Figure 2 for an illustration, and Figure 3 for a comparison of Theorem 6 with Theorem 3.

Theorem 6, like its counterpart Theorem 3, implies that provably $d_{\mathrm{i}}^{\mathbf{1}}$-stable invariants computed from our estimators approximate the corresponding invariants defined on their target, which this time is $\mathrm{Lan}_\iota H_*(f)$. And since the interleaving is now vertical – hence stronger than an ordinary interleaving according to Remark 1, we can get more precise statements for some invariants. For instance, the induced bottleneck matching on multigraded Betti numbers is vertical, as proven in the full version [1] and illustrated in Figure 2.

4 Estimators Computation

Our main estimators, $H_*(\mathcal{R}^{\delta\to 2\delta}(f|_P))$ and $H_*(\mathcal{R}^{\bullet\to 2\bullet}(f|_P))$, derive from the function Rips filtration $\mathcal{R}^\bullet(f|_P)$, whose computation is standard – see [1, Section 7] for details. Here we focus on the computation of invariants once $\mathcal{R}^\bullet(f|_P)$ has been built. The difficulty

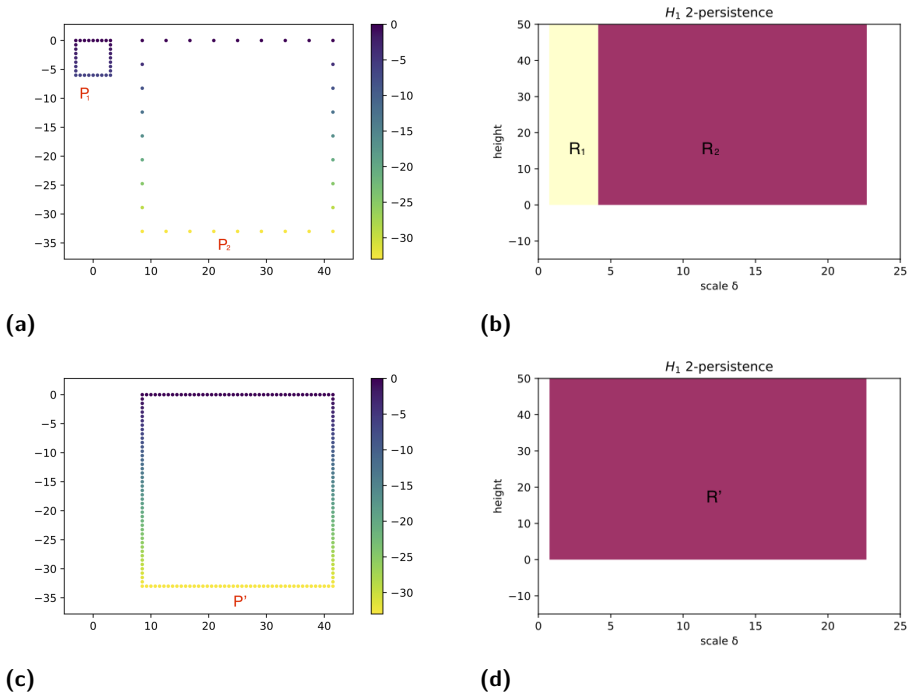


Figure 3 Contrasting Theorem 6 with Theorem 3. **(a)**: The input is $P = P_1 \sqcup P_2$, where P_1 and P_2 are two point clouds regularly sampled from two disjoint squares X_1, X_2 in the plane. Distances within each square are shortest-path distances along the boundary, while distances between squares are infinite. Here $\mathscr{f}$ is the vertical height function, and its persistent homology in degree 1 is considered. **(b)**: The estimator $H_1(\mathcal{R}^{\bullet\to 2\bullet}(\mathscr{f}|_P))$ is an interval-decomposable module whose summands have half-open rectangle supports, respectively R_1 (in yellow) and R_2 (in red). The scalings of X_1, X_2, and their respective sampling densities, have been adjusted so that $R_1 \cap R_2 = \emptyset$ while $R_1 \cup R_2$ is a single rectangle R' shown in subfigure **(d)**. In turn, the interval module with support R' can be realized as the degree 1 persistent homology of another sample P' from a square in the plane, shown in subfigure **(c)**. Theorem 6 guarantees that $H_1(\mathcal{R}^{\bullet\to 2\bullet}(\mathscr{f}|_P))$ is interleaved with $\mathrm{Lan}_\iota H_1(\mathscr{f}|_{X_1})$ over R_1 and with $\mathrm{Lan}_\iota H_1(\mathscr{f}|_{X_2})$ over R_2, leading to the two summands in **(b)**. By contrast, Theorem 3 only guarantees interleavings within the vertical slices, which is not sufficient to discriminate the module with two summands in **(b)** from the module with a single summand in **(d)**.

is that our estimators do not come from a single filtration, but from the inclusion of two nested filtrations sitting inside $\mathcal{R}^\bullet(\mathscr{f}|_P)$. For some invariants, such as the Hilbert function or rank invariant, existing algorithms adapt straightforwardly [5]. However, for minimal free presentations – from which many invariants, including multigraded Betti numbers, derive – there is no efficient existing algorithm, except in special situations such as when homology in degree 0 is considered, see the full version [1, Section 7.2.3] for details.

To address this, we provide an algorithm for the general problem of computing a free presentation of the image of any morphism of finitely presented persistence modules $\tau : M \to N$, and then we apply it to the special case of $H_*(\mathcal{R}^{\bullet\to 2\bullet}(\mathscr{f}|_P))$. Here we merely outline the mathematical construction of a free presentation of $\mathrm{Im}(\tau)$, which explains our approach. The algorithmic details and pseudocode are provided in [1, Section 7]. Note that a simpler variant to obtain a free presentation of the cokernel of τ appeared in [12, Proposition 4.14].

Let $\tau : M \to N$ be a morphism between finitely presented persistence modules, with respective free presentations $p_1 : P_1 \to P_0$ and $q_1 : Q_1 \to Q_0$. Since P_0 is projective, we can lift τ to a map $\gamma : P_0 \to Q_0$ so that the following diagram commutes:

$$\begin{array}{ccccccccc} P_1 & \xrightarrow{p_1} & P_0 & \xrightarrow{p_0} & M & \longrightarrow & 0 \\ & & \downarrow{\scriptstyle \gamma} & & \downarrow{\scriptstyle \tau} & & \\ Q_1 & \xrightarrow{q_1} & Q_0 & \xrightarrow{q_0} & N & \longrightarrow & 0. \end{array} \tag{2}$$

From this diagram, we can derive a free presentation of $\mathrm{Im}(\tau)$ as follows. Consider the pullback P_1' of the subdiagram $P_0 \xrightarrow{\gamma} Q_0 \xleftarrow{q_1} Q_1$, i.e., the kernel of the morphism $P_0 \oplus Q_1 \xrightarrow{\gamma - q_1} Q_0$, where $(\gamma - q_1)(u, v) := \gamma(u) - q_1(v)$ for any $(u, v) \in P_0 \oplus Q_1$. Let $\pi_0 : P_1' \to P_0$ and $\pi_1 : P_1' \to Q_1$ denote the canonical projections from $P_1' \subseteq P_0 \oplus Q_1$ to P_0 and Q_1, respectively. Next, let P_2' be the free cover of P_1', yielding the surjection $\beta : P_2' \twoheadrightarrow P_1'$. We illustrate these morphisms in the following commutative diagram, where P_1', P_0, Q_0, Q_1 form a pullback square:

$$\begin{array}{ccccccccc} P_2' & \xrightarrow{\beta} & P_1' & \overset{\pi_0}{\longrightarrow} & & & & & \\ & \searrow^{\pi_1} & & & & & & & \\ & & & & P_1 & \xrightarrow{p_1} & P_0 & \xrightarrow{p_0} & M & \longrightarrow & 0 \\ & & & & & & \downarrow{\scriptstyle \gamma} & & \downarrow{\scriptstyle \tau} & & \\ & & & & Q_1 & \xrightarrow{q_1} & Q_0 & \xrightarrow{q_0} & N & \longrightarrow & 0 \end{array} \tag{3}$$

This diagram yields a free presentation of $\mathrm{Im}(\tau)$ as follows (proof is in the full version [1]):

▶ **Proposition 7.** *The sequence $P_2' \xrightarrow{\pi_0 \circ \beta} P_0 \xrightarrow{\eta} \mathrm{Im}(\tau) \to 0$ is exact, where η comes from the factorization of $\tau \circ p_0$ through its image $\mathrm{Im}(\tau)$. Consequently, the morphism $\pi_0 \circ \beta : P_2' \to P_0$ is a free presentation of $\mathrm{Im}(\tau)$, and in fact a finite free presentation.*

5 Experiments

In this section, we illustrate our approximation and convergence results through practical experiments on both synthetic and real-world datasets.

5.1 Toy examples

We validate the claims in Theorems 3 and 6 through a simple example consisting of two circles with distinct radii and concentration levels, as shown in Figure 4 (a). In Figure 4 (b) we analyze the error between the estimator $H_1(\mathcal{R}^{\delta \to 2\delta}(f|_P))$ and the target $H_1(f)$ as a function of the scale parameter δ. As guaranteed by Theorem 3, for δ within the range $[2\varepsilon, \varrho_X/2)$ the error between the estimator and the target is small, controlled by δ. The approximation remains good even for δ going beyond that range, which is not explained by the theory but occurs here because the space and function are very regular in this example.

In Figure 4 (c) we show a visualization of our estimator $H_1(\mathcal{R}^{\bullet \to 2\bullet}(f|_P))$ using `MMA` [17]. As expected from Theorem 6, we observe two topological features in our estimator, corresponding to the two circles in the dataset. The values of δ at which these features appear then disappear are aligned with the endpoints of the interval of values of δ for which $H_1(\mathcal{R}^{\delta \to 2\delta}(f|_P))$ is a good approximation of $H_1(f)$ in Figure 4 (b).

For comparison, in Figure 4 (d) we show a visualization of the estimator $H_1(\mathcal{C}^{\bullet}(f|_P))$ using `MMA`, for which we use the Euclidean distance instead of the geodesic distance in order to enable the construction of the function–Čech filtration. Again, we see two topological features corresponding to the two circles, and $H_1(\mathcal{C}^{\delta}(f|_P))$ is a good approximation of $H_1(f)$ for δ within a fairly large range.

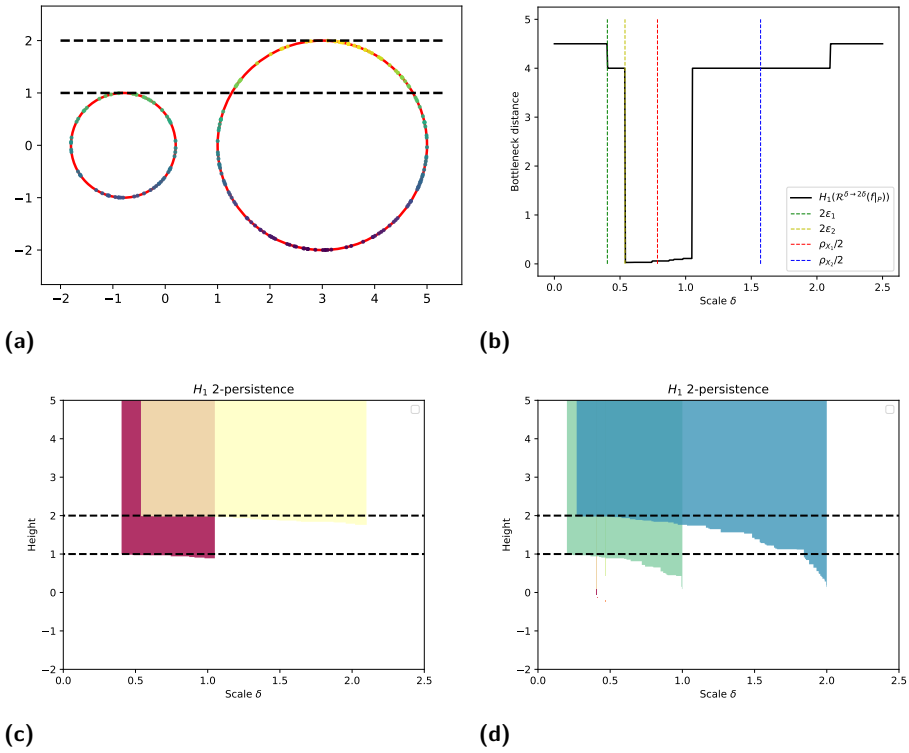


Figure 4 (a): A sample P from a space $X = X_1 \sqcup X_2$ composed of two circles equipped with geodesic distances, each uniformly sampled with distinct radii and concentration levels. The target function $f\colon X \to \mathbb{R}$ is the height function on the two circles. The barcode of the target $H_1(f)$ consists of two infinite bars, each originating from a level marked as a dashed line. **(b)**: Bottleneck distance between the barcode of the estimator $H_1\left(\mathcal{R}^{\delta\to 2\delta}(f|_P)\right)$ and that of the target $H_1\left(f\right)$ as a function of δ. Here, ε_1 (resp. ε_2) is the sampling error of X_1 (resp. X_2), and ϱ_{X_1} (resp. ϱ_{X_2}) the convexity radius of X_1 (resp. X_2). All infinite bars are truncated at 10 to ensure the bottleneck distances are finite. **(c)**: Visualization of the estimator $H_1\left(\mathcal{R}^{\bullet\to 2\bullet}(f|_P)\right)$ computed using `MMA`. **(d)**: Visualization of the estimator $H_1\left(\mathcal{C}^{\bullet}(f|_P)\right)$ computed using `MMA`. Each colored region represents a persistent topological feature. Dashed lines indicate birth times of bars in the barcode of $H_1(f)$.

We now add noise to the dataset sampled from the two circles, as shown in Figure 5 (a). As proven in the full version [1, Section 3.3], the trend is the same. While the theory requires a factor $\alpha > 2$ in $H_1(\mathcal{R}^{\bullet\to\alpha\bullet}(f|_P))$ to handle metric noise, Figure 5 shows that, in this example, the estimator $H_1(\mathcal{R}^{\bullet\to 2\bullet}(f|_P))$ performs well even with $\alpha = 2$.

5.2 Real dataset of immune cells (involving Lipschitz functions)

In this section, we consider the digitized immunohistochemistry dataset from [24]. This dataset is comprised of manually annotated cell locations from three cell types, namely cytotoxic T lymphocytes, regulatory T lymphocytes, and macrophages, each characterized by the expression of one of the proteins CD8, FoxP3, and CD68, respectively – see Figure 6.

The index $i \in \{1, 2, 3\}$ refers to the protein in the ordered set $\{\text{CD8}, \text{FoxP3}, \text{CD68}\}$, and $f_i\colon [0,1]^2 \to \mathbb{R}$ denotes the negative density estimate associated with that protein. Since the functions f_i are unknown, we estimate the density of each point cloud using kernel density

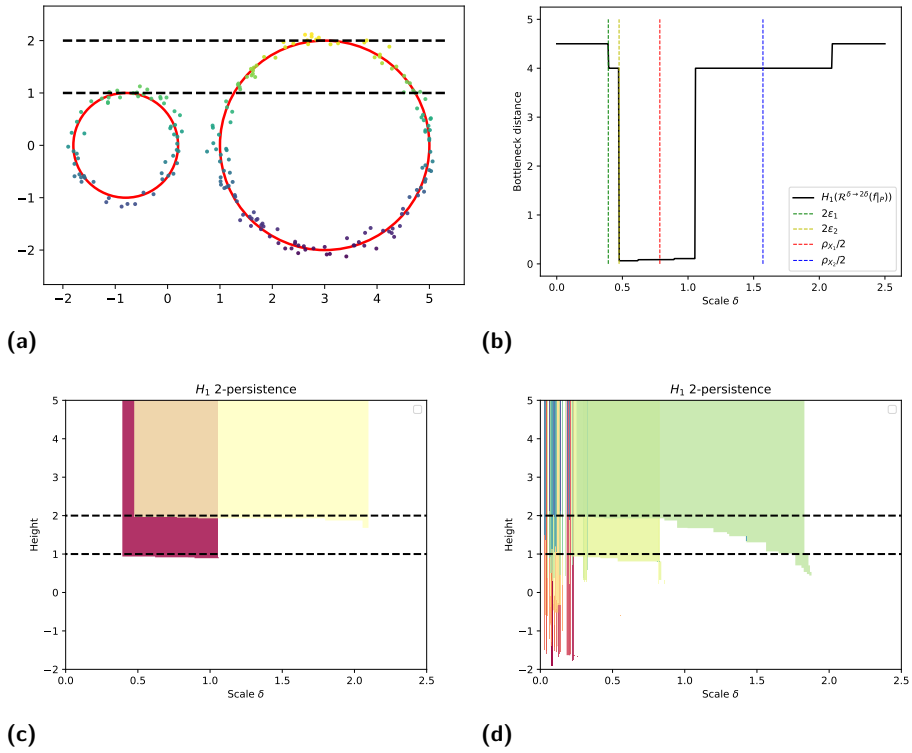


Figure 5 A noisy analog of Figure 4.

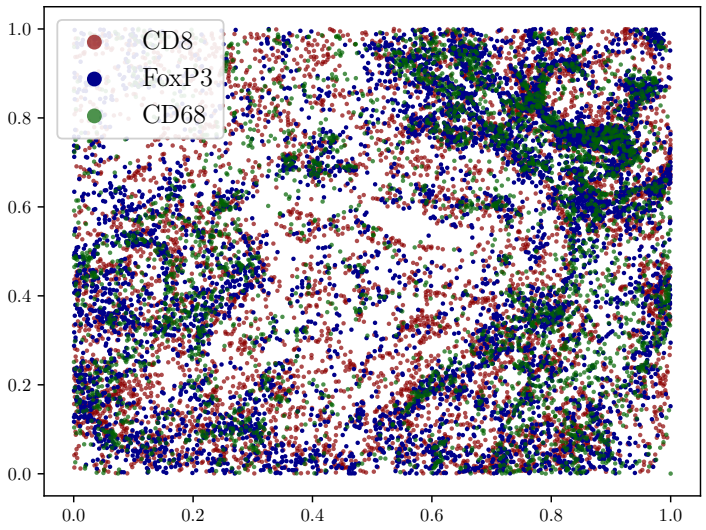
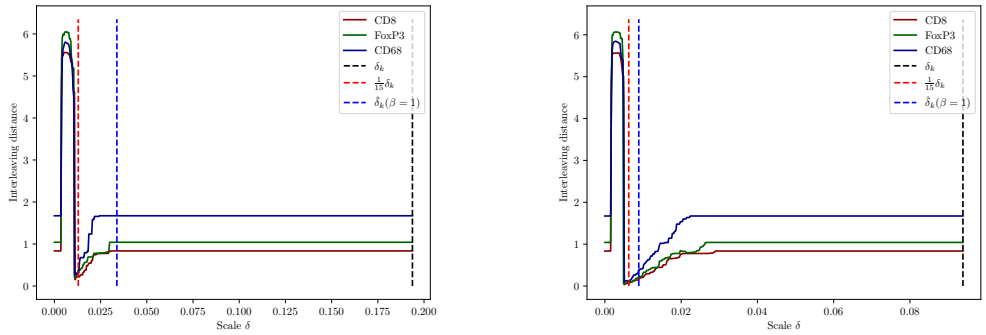


Figure 6 Point cloud dataset, with colors indicating the protein labels CD8, FoxP3, and CD68.



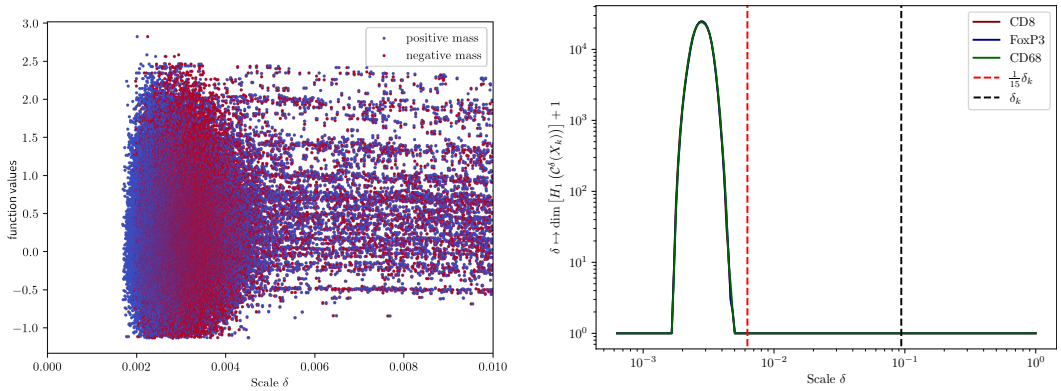
(a) Errors for $H_1(\mathcal{C}^\delta(\mathscr{f}_i|_{X_{10\ 000}}))$.

(b) Errors for $H_1(\mathcal{C}^\delta(\mathscr{f}_i|_{X_{50\ 000}}))$.

Figure 7 Interleaving distance between our estimators and the target $H_1(\mathscr{f})$ w.r.t. the scale δ.

estimation, which yields approximations – also called $\mathscr{f}_1, \mathscr{f}_2, \mathscr{f}_3$ for simplicity – that are Lipschitz continuous by construction. We then discretize the domain $[0,1]^2$ on a grid (using a resolution of $1\,000 \times 1\,000$ for Section 5.2.1 and of 500×500 for Section 5.2.2) and compute the persistence module of the cubical complex induced by the estimated function on this grid, which serves as a proxy for the ground-truth target.

In the following experiments, in order to reach smaller Hausdorff distances between the sample X_k and the domain $X = [0,1]^2$, we first draw 10,000 points uniformly from X, and then select X_k using k-means farthest point sampling applied to this set. In practice, the choice of the rate δ_k prescribed in Theorem 5 can be improved significantly when the sample X_k is well-behaved. We propose heuristics to estimate good rates in non-asymptotic regimes.



(a) Hilbert signed bars of $H_1(\mathcal{C}^\bullet(\mathscr{f}_1|_{X_k}))$.

(b) Pointwise dimension of $H_1(\mathcal{C}^\delta(X_k))$ w.r.t δ.

Figure 8 Heuristic to determine a good multiplicative constant for the rate δ_k, where $k = 50\,000$.

5.2.1 Estimating a single function $\mathscr{f}_i$

We empirically validate our theoretical results by examining the estimators $H_1(\mathcal{C}^\delta(\mathscr{f}_i|_{X_k}))$ with $i \in \{1,2,3\}$ built from the three associated point clouds. We first focus on the convergence rate of the estimators as the sample size k increases, with respect to the choice of scale parameter δ_k. From Theorem 5, since the sampling measure is known – namely the uniform measure on $[0,1]^2$, which is (a,b)-standard with parameters $a = \frac{\pi}{4}$ and $b = 2$ – we

can derive an asymptotic theoretical rate $\delta_k = 4\left(\frac{8\log(k)}{\pi k}\right)^{\frac{1}{2}}$. However, as shown in Figure 7, this choice of δ_k is very conservative, and consequently the estimators $H_1(\mathcal{C}^{\delta_k}(\ell_i|_{X_k}))$ do not perform better than the zero module. This is explained by the fact that δ_k is designed for worst-case sampling measures and asymptotic regimes. As shown in Figure 7, the estimators $H_1(\mathcal{C}^{\delta_k}(\ell_i|_{X_k}))$ achieve good performance only when δ_k is smaller than 0.02, which would require sample sizes k larger than 10^6.

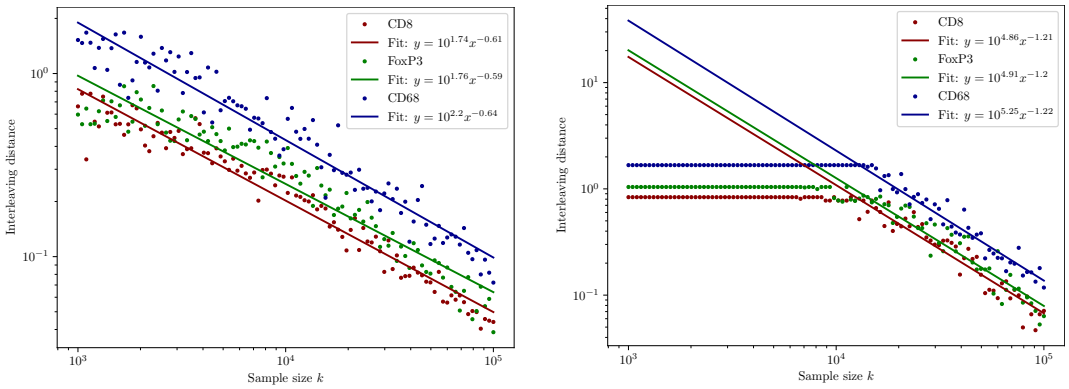
Another option is to consider the estimated rate $\hat{\delta}_k$ mentioned in Theorem 5, defined as $\hat{\delta}_k := d_{\mathrm{H}}(X_{s_k}, X_k)$ where $X_{s_k} \subseteq X_k$ consists of the first s_k sample points of X_k. In Figure 7, we observe that this rate is far less conservative than δ_k and already exhibits its asymptotic behavior when k is larger than 50 000, achieving good performance. In this example though, the rate $\hat{\delta}_k$ has some limitations:

- the bad behavior of the corresponding estimators on small sample sizes ($k \leq 10^4$);
- the dependence of $\hat{\delta}_k$ on a parameter β, which needs to be tuned;
- the fact that the knowledge of the ambient dimension and the probability measure from which the points are sampled are not leveraged.

Here is a heuristic we apply to select a relevant rate δ'_k. Since the dimension of the domain is known, we set δ'_k to be a fixed fraction of the theoretical rate δ_k. This choice is motivated in the full version [1] and ensures that, up to a multiplicative constant, the rate $(1/k)^{1/d}$ cannot be improved for any (possibly deterministic) k-sample of a d-dimensional space. To determine a good multiplicative constant, we notice from Figure 7 that, when $\delta \in [0, d_{\mathrm{H}}(X_k, X))$, the error of the estimators $H_1(\mathcal{C}^{\delta}(\ell_i|_{X_k}))$ soars, due to the fact that $\mathcal{C}^{\delta}(X_k)$ and X do not have the same topology. According to Theorem 3, the optimal scale δ is then given as the smallest value lying above that interval, that is: $d_{\mathrm{H}}(X_k, X)$. Our rate δ'_k is an estimate of this value. In order to compute it, we look at the topological changes of $\mathcal{C}^{\delta}(X_k)$ as δ increases. Since the measure is uniform and thus does not exhibit pathological local behaviors, the artifacts in the topological type of $\mathcal{C}^{\delta}(X_k)$ should all vanish roughly at the same scale, and slightly before $d_{\mathrm{H}}(X_k, X)$. This phenomenon, suggested by the shape of the curves in Figure 7, is further illustrated in Figure 8(a), where we consider the *Hilbert function signed barcode* [18, 21] of $H_1(\mathcal{C}^{\bullet}(\ell_1|_{X_k}))$: after an initial transition phase ($\delta < 0.005$) during which a concentration of positive generators appears, followed then by a concentration of negative generators, the structure of the module stabilizes as these positive and negative generators cancel each other out. This is highlighted when summing the signed generators within the half-plane $(-\infty, \delta] \times \mathbb{R}$, or equivalently, when computing the pointwise dimension of $H_1(\mathcal{C}^{\delta}(X_{50\ 000}))$, for δ ranging from 0 to $+\infty$, as shown in Figure 8(b). Since the convexity radius ϱ_X is infinite in this case, we have $H_1(\mathcal{C}^{\delta}(X_k)) \cong H_1(\mathcal{O}^{\delta}(X_k)) \cong H_1(X)$ for all $\delta > d_{\mathrm{H}}(X_k, X)$, and so our heuristic to determine a suitable rate $(\delta'_k)_{k\in\mathbb{N}}$ from a sample X_{k_0} for some sufficiently large $k_0 \in \mathbb{N}$ is:

1. Compute the pointwise dimension of $H_1(\mathcal{C}^{\delta}(X_{k_0}))$ as in Figure 8(b) for a range of δ;
2. Identify the transition phase of $\dim H_1(\mathcal{C}^{\delta}(X_{k_0}))$;
3. Choose a scale δ^* past this phase, so that $\delta \in [\delta^*, \infty) \mapsto \dim H_1(\mathcal{C}^{\delta}(X_{k_0}))$ is constant;
4. Define the rate $(\delta'_k)_k$ for $k \in \mathbb{N}$ as $\delta'_k := \frac{\delta^*}{\delta_{k_0}}\delta_k$.

In Figure 9, we observe that both rates δ'_k and $\hat{\delta}_k$ achieve good performance asymptotically, with a convergence rate that is at least of the order of $\left(\frac{\log(k)}{k}\right)^{1/2}$. However, the asymptotic regime is reached earlier with δ'_k than with $\hat{\delta}_k$ – while, as we said, it is not reached with δ_k.



(a) Convergence rate with $\delta'_k = \frac{1}{15}\delta_k$. **(b)** Convergence rate with $\hat{\delta}_k$.

Figure 9 Interleaving distance between the estimators and the target w.r.t. the sample size k.

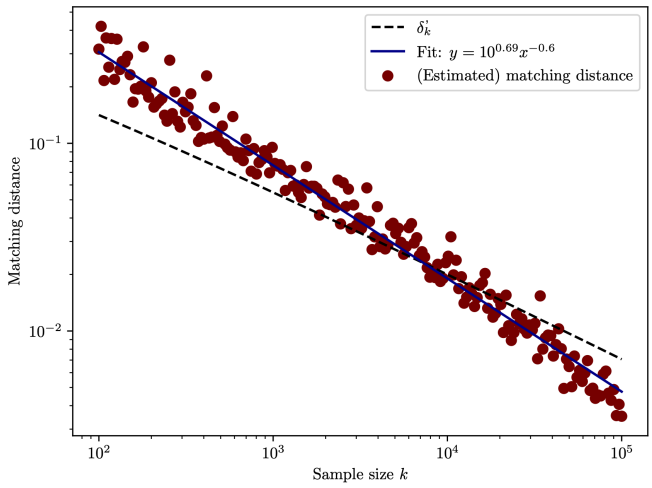


Figure 10 Estimated matching distance between the estimator $H_0(\mathcal{R}^{\delta'_k \to 2\delta'_k}(\mathcal{f}|_{X_k}))$ and the ground truth $H_0(\mathcal{f})$ as a function of the sample size k under random sampling.

5.2.2 Estimating the combined functions $(\mathcal{f}_1, \mathcal{f}_2, \mathcal{f}_3)$

We now study the function $\mathcal{f} = (\mathcal{f}_1, \mathcal{f}_2, \mathcal{f}_3)\colon [0,1]^2 \to \mathbb{R}^3$ and approximate the corresponding 3-parameter persistence module $H_0(\mathcal{f})$. For this we employ the estimator $H_0(\mathcal{R}^{\delta'_k \to 2\delta'_k}(\mathcal{f}|_{X_k})) \cong H_0(\mathcal{R}^{2\delta'_k}(\mathcal{f}|_{X_k}))$, where, for each sample size k, the scale parameter is given by the same δ'_k as in Section 5.2.1. Since computing the interleaving distance in this context is NP-hard [3], we use the matching distance [15] as a proxy, estimated via Monte Carlo with $1,000$ random lines in $\mathbb{R}^3$. The results are shown in Figure 10. Notably, the observed convergence rate matches that of the previous experiment in Figure 9(a), which suggests that the empirical convergence rate is independent of the number of parameters.

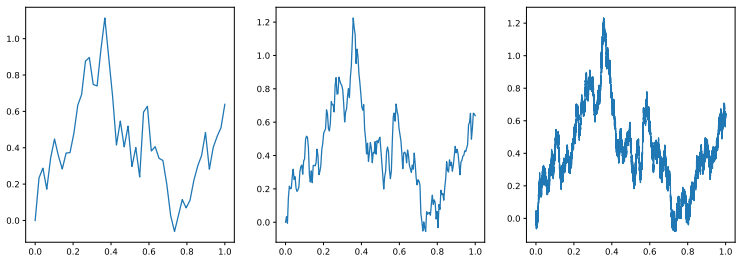


Figure 11 Samplings of a Brownian motion on $[0,1]$ with, respectively, 50, 200, and 10^6 points.

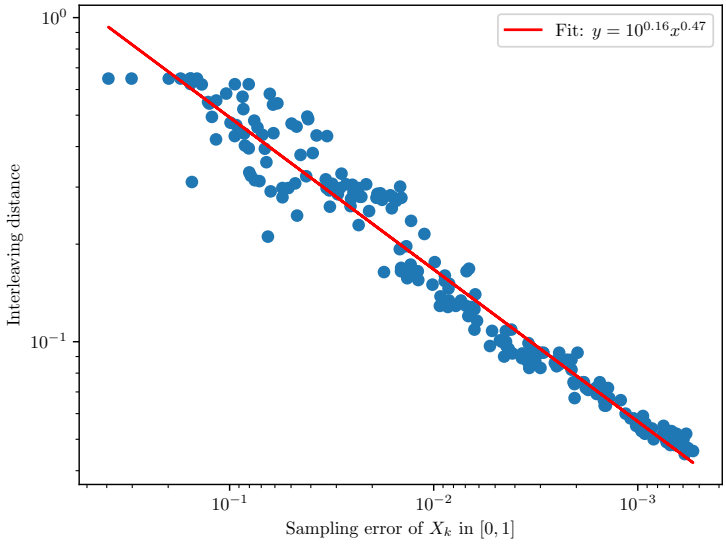


Figure 12 Log–log plot of the convergence of $H_0(\mathcal{C}^{\varepsilon_k}(f|_{X_k}))$ toward $H_0(f)$ for a Brownian motion f. The x-axis represents the sampling error $\varepsilon_k = d_{\mathrm{H}}(X_k, [0,1])$, and the y-axis represents the interleaving distance between $H_0(\mathcal{C}^{\varepsilon_k}(f|_{X_k}))$ and $H_0(f)$.

5.3 Brownian motion in 1-d (involving a Hölder function)

The Brownian motion is not Lipschitz continuous but almost surely almost everywhere ω-continuous for any modulus of continuity ω of the form $\omega\colon x \mapsto c\,x^{\frac{1}{2}-\varepsilon}$, with $\varepsilon > 0$ and $c > 0$ large enough [20, Corollaries 1.20 and 5.3]. Let $f : [0,1] \to \mathbb{R}$ be a realization of a standard Brownian motion, which can be approximated by a Rademacher random walk.

For each $k \in \mathbb{N}$, we build a k-sample X_k in $[0,1]$ according to the uniform distribution, as shown in Figure 11 for several values of k. Then, since our goal here is to illustrate Theorem 3, we assume we have access to the sampling error $\varepsilon_k := d_{\mathrm{H}}(X_k, [0,1])$ and so we use the estimator $H_0(\mathcal{C}^{\varepsilon_k}(f|_{X_k}))$ to approximate the target $H_0(f)$ for various sample sizes k. The corresponding results are shown in Figure 12. A regression analysis of the experimental data indicates that $d_{\mathrm{i}}^{\mathbf{1}}\left(H_0(\mathcal{C}^{\varepsilon_k}(f|_{X_k})), H_0(f)\right) \approx 10^{0.16}\,{\varepsilon_k}^{0.47}$, which aligns closely with the theoretical prediction from Theorem 3, stating that the interleaving distance between $H_0(\mathcal{C}^{\varepsilon_k}(f|_{X_k}))$ and $H_0(f)$ should be at most $\omega(\varepsilon_k) \lesssim {\varepsilon_k}^{1/2}$.

5.4 Timings

In the following table we report the running times for some of our experiments. Only the first one involves our algorithm for computing a presentation of $H_*(\mathcal{R}^{\bullet\to 2\bullet}(\mathscr{f}|_P))$. For all computations we used an Intel® Core™ i9-12900K CPU (5.2 GHz max) equipped with 125 GB of RAM.

Experiment	Estimator	#Params	#Points	Filt. size	Filt. time	Invariant time
2 circles	$H_1(\mathcal{R}^{\bullet\to 2\bullet}(\mathscr{f}\|_P))$	2	220	13,110	11.8 ms	6.5 ms (presentation)
Immune cells (single $\mathscr{f}_i$)	$H_1(\mathcal{C}^\bullet(\mathscr{f}_i\|_{X_k}))$	2	6172	24,370	864 ms	50.3 ms (Hilbert measure)
Brownian motion	$H_0(\mathcal{C}^{\varepsilon_k}(\mathscr{f}\|_{X_k}))$	1	10,000	37,359	1.21 s	5.4 ms (barcode)

References

1 Ethan André, Jingyi Li, David Loiseaux, and Steve Oudot. Estimating the persistent homology of $\mathbb{R}^n$-valued functions using function-geometric multifiltrations, 2025. arXiv:2412.04162.

2 Ulrich Bauer and Michael Lesnick. Induced matchings and the algebraic stability of persistence barcodes. *Journal of Computational Geometry*, 6, 2015. doi:10.20382/JOCG.V6I2A9.

3 Håvard Bakke Bjerkevik, Magnus Bakke Botnan, and Michael Kerber. Computing the interleaving distance is np-hard. *Foundations of Computational Mathematics*, 20(5):1237–1271, 2020. doi:10.1007/S10208-019-09442-Y.

4 Omer Bobrowski, Sayah Mukherjee, and Jonathan E. Taylor. Topological consistency via kernel estimation. *Bernoulli*, 23(1):288–328, 2017.

5 Magnus Bakke Botnan, Vadim Lebovici, and Steve Oudot. On rectangle-decomposable 2-parameter persistence modules. *Discrete & Computational Geometry*, pages 1–24, 2022.

6 Mathieu Carriere, Bertrand Michel, and Steve Oudot. Statistical analysis and parameter selection for mapper. *Journal of Machine Learning Research*, 19(12):1–39, 2018. URL: https://jmlr.org/papers/v19/17-291.html.

7 F. Chazal and S. Y. Oudot. Towards persistence-based reconstruction in Euclidean spaces. In *Proc. 24th ACM Sympos. Comput. Geom.*, pages 232–241, 2008.

8 Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J Guibas, and Steve Y Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 237–246, 2009. doi:10.1145/1542362.1542407.

9 Frédéric Chazal, Vin De Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*, volume 10. Springer, 2016. doi:10.1007/978-3-319-42545-0.

10 Frédéric Chazal, Leonidas J Guibas, Steve Y Oudot, and Primoz Skraba. Scalar field analysis over point cloud data. *Discrete & Computational Geometry*, 46(4):743–775, 2011. doi:10.1007/S00454-011-9360-X.

11 Frédéric Chazal, Leonidas J Guibas, Steve Y Oudot, and Primoz Skraba. Persistence-based clustering in Riemannian manifolds. *Journal of the ACM (JACM)*, 60(6):1–38, 2013. doi:10.1145/2535927.

12 Tamal K Dey, Jan Jendrysiak, and Michael Kerber. Decomposing multiparameter persistence modules. *arXiv preprint*, 2025. arXiv:2504.08119.

13 Allen Hatcher. *Algebraic Topology*. Cambridge Univ. Press, 2001. URL: http://www.math.cornell.edu/~hatcher/.

14 Rafal Komendarczyk, Sushovan Majhi, and Will Tran. Topological stability and latschev-type reconstruction theorems for spaces of curvature bounded above. *arXiv preprint*, 2024. arXiv:2406.04259.

15 Claudia Landi. The rank invariant stability via interleavings. In *Research in computational topology*, pages 1–10. Springer, 2018.

16 Janko Latschev. Vietoris-Rips complexes of metric spaces near a closed Riemannian manifold. *Archiv der Mathematik*, 77(6):522–528, 2001.

17 David Loiseaux, Mathieu Carrière, and Andrew J. Blumberg. Multi-parameter Module Approximation: An efficient and interpretable invariant for multi-parameter persistence modules with guarantees. *Journal of Applied and Computational Topology*, 9(4):26, December 2025. doi:10.1007/s41468-025-00222-y.

18 David Loiseaux, Luis Scoccola, Mathieu Carrière, Magnus Bakke Botnan, and Steve Oudot. Stable Vectorization of Multiparameter Persistent Homology using Signed Barcodes as Measures. *Advances in Neural Information Processing Systems*, 36:68316–68342, December 2023.

19 Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

20 Peter Mörters and Yuval Peres. *Brownian Motion*, volume 30. Cambridge University Press, 2010.

21 Steve Oudot and Luis Scoccola. On the stability of multigraded Betti numbers and Hilbert functions. *SIAM Journal on Applied Algebra and Geometry*, 8(1):54–88, 2024. doi:10.1137/22M1489150.

22 Steve Oudot and Lukas Waas. A persistent version of Latschev's theorem. In *42nd International Symposium on Computational Geometry (SoCG 2026)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 82:1–82:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. doi:10.4230/LIPIcs.SoCG.2026.82.

23 Alexander Rolle and Luis Scoccola. Stable and consistent density-based clustering via multiparameter persistence. *Journal of Machine Learning Research*, 25(258):1–74, 2024.

24 Oliver Vipond, Joshua A. Bull, Philip S. Macklin, Ulrike Tillmann, Christopher W. Pugh, Helen M. Byrne, and Heather A. Harrington. Multiparameter persistent homology landscapes identify immune cell spatial patterns in tumors. *Proceedings of the National Academy of Sciences*, 118(41):e2102166118, October 2021. doi:10/gpd543.

Computing L_∞ Hausdorff Distances Under Translations: The Interplay of Dimensionality, Symmetry and Discreteness

Sebastian Angrick
Karlsruhe Institute of Technology, Germany

Kevin Buchin
TU Dortmund, Germany

Geri Gokaj
Karlsruhe Institute of Technology, Germany

Marvin Künnemann
Karlsruhe Institute of Technology, Germany

Abstract

To measure the similarity of the *shape* of point sets, rather than their mere closeness in space, various notions of a *Hausdorff distance under translation* have been investigated. Specifically, let P and Q denote point sets of n and m points, respectively, in $\mathbb{R}^d$. We consider the task of computing the minimum distance $d(P, Q + \tau)$ over an admissible set of translations $\tau \in T$, where $d(\cdot, \cdot)$ denotes the Hausdorff distance under the L_∞-norm. As variants, we distinguish between *continuous* ($T = \mathbb{R}^d$) or *discrete* (T is a given finite set of t translations) as well as *directed* or *undirected* (choosing the directed or undirected Hausdorff distance for $d(\cdot, \cdot)$).

We seek to apply the paradigm of fine-grained complexity to understand the complexity of these variants, and in particular: How is the running time influenced by the dimension d, the relationship between n and m, and the specific choice of variant? As our main results, we obtain:

- The asymmetric definition of the most studied variant, the continuous directed Hausdorff distance, results in an *intrinsically asymmetric* time complexity: While (Chan, SoCG'23) established a symmetric $\tilde{O}((nm)^{d/2})$ upper bound for all $d \geq 3$ and proved it to be conditionally optimal for *combinatorial* algorithms whenever $m \leq n$, we show that this lower bound does not hold for the case $n \ll m$, by providing a combinatorial, almost-linear-time algorithm for $d = 3$ and $n = m^{o(1)}$. We further prove *general*, i.e., non-combinatorial, conditional lower bounds for $d \geq 3$, in particular: (1) $m^{\lfloor d/2 \rfloor - o(1)}$ for small n and (2) $n^{d/2-o(1)}$ for $d = 3$ and small m.
- We observe that the directed and undirected case is closely related, in particular, all our lower bounds for $d \geq 3$ hold for both the directed and undirected variant. A remarkable exception is the case of $d = 1$ for which we provide a conditional separation. Specifically, in contrast to the undirected variants being solvable in near-linear time (Rote, IPL'91), we show that the directed variants are at least as hard as the additive problem MaxConv LowerBound introduced in (Cygan, Mucha, Wegrzycki and Wlodarczyk, TALG'19).
- We show that the discrete variants reduce to a variant of 3SUM for $d \leq 3$. This gives a barrier in proving a tight lower bound of these variants under the Orthogonal Vectors Hypothesis (OVH); in contrast, the continuous variants admit a tight conditional lower bound under OVH in $d = 2$ (Bringmann, Nusser, JoCG'21).

These results reveal an intricate interplay of dimensionality, symmetry and discreteness in determining the fine-grained complexity of computing Hausdorff distances under translation.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Computational geometry

Keywords and phrases Hausdorff Distance, Fine-Grained Complexity, Computational Geometry, Translation-Invariant Similarity Measures

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.7

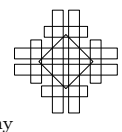


© Sebastian Angrick, Kevin Buchin, Geri Gokaj, and Marvin Künnemann;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Computational Geometry (SoCG 2026).
Editors: Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri; Article No. 7; pp. 7:1–7:15



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Related Version *Full Version*: https://arxiv.org/pdf/2603.08890

Funding *Geri Gokaj*: Partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 462679611.
Marvin Künnemann: Partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 462679611.

Acknowledgements The authors would like to thank Jean Cardinal, Anita Dürr, Nick Fischer, Virginia Vassilevska Williams, and Karol Węgrzycki for stimulating discussions and contributions during the early stages of this work, as well as the Lorentz center and the organizers of its workshop on *Fine-Grained & Parameterized Computational Geometry* for facilitating these discussions.

1 Introduction

Consider the classic Hausdorff distance of given point sets P and Q, which comes in two flavors: the *directed* Hausdorff distance $\delta_{\overrightarrow{H}}(P,Q) \coloneqq \max_{p\in P} \min_{q\in Q} \|p-q\|$, as well as the *undirected* Hausdorff distance $\delta_H(P,Q) \coloneqq \max\{\delta_{\overrightarrow{H}}(P,Q), \delta_{\overrightarrow{H}}(Q,P)\}$. Here and throughout the paper, we fix our metric space to be $(\mathbb{R}^d, L_\infty)$. The two variants are sometimes also referred to as one-sided/bidirectional or one-way/two-way. Furthermore, δ_H yields a metric, in contrast to $\delta_{\overrightarrow{H}}$. For any constant $d \in \mathbb{N}$ and point sets P and Q of n and m points in $\mathbb{R}^d$, we can compute these distance in near-linear time[1] $\tilde{O}(n+m)$ using L_∞-nearest neighbor search, e.g., in a range tree [18].

A computationally more challenging task is to compute the Hausdorff distance under some transformation, i.e., to minimize the Hausdorff distance of P' and Q among all sets P' obtained from P using a set of admissible transformations. Such notions are well-studied from the perspective of *shape matching* or *geometric pattern matching*, see, e.g. [3, 4], where P and Q model geometric objects and we are interested in comparing their general shapes rather than their mere closeness in space. In this paper, we focus on arguably one of the most basic settings, i.e., computing L_∞ Hausdorff distances under translation. Specifically, when T denotes a set of admissible translations, the Hausdorff distance under translation is defined as $\min_{\tau\in T} \delta(P+\tau, Q)$ where δ denotes either the directed or undirected Hausdorff distance, i.e., $\delta \in \{\delta_{\overrightarrow{H}}, \delta_H\}$.

The most popular setting is $T = \mathbb{R}^d$, which yields the natural translation-invariant version of the Hausdorff distance; we refer to this as the **(continuous) Hausdorff distance under translation (HuT)**. This measure has received significant interest from the computational geometry community both for L_∞ and other norms [20, 16, 17, 21, 11, 10, 5, 7, 9], yet only for $d = 2$ tight upper and conditional lower bounds are known (see below for details).

Another natural version is to let T be a finite set given as part of the input. We refer to this as the **discrete Hausdorff distance under translation (DiscHuT)**. This version occurs naturally in the context of approximating the (continuous) Hausdorff distance under translation. More precisely, computing an α-approximation for the continuous version can be achieved by choosing $f(\alpha, d)$ many translations T such that $\min_{\tau\in T} \delta_{\overrightarrow{H}}(P+\tau, Q) \leq \alpha \min_{\tau\in\mathbb{R}^d} \delta_{\overrightarrow{H}}(P+\tau, Q)$. Since we can compute the discrete Hausdorff distance under translation in time $\tilde{O}(|T|(|P|+|Q|))$, this yields baseline α-approximation algorithms for the continuous Hausdorff distance under translation with running time $\tilde{O}(f(\alpha,d)(n+m))$; see also [15, 22]. Consequently, obtaining faster algorithms for the discrete Hausdorff distance

[1] Throughout the paper, we use $\tilde{O}(\cdot)$, to hide polylogarithmic terms in n and m. Since we assume d to be constant, this includes factors of the form $\log^{O(d)} nm$.

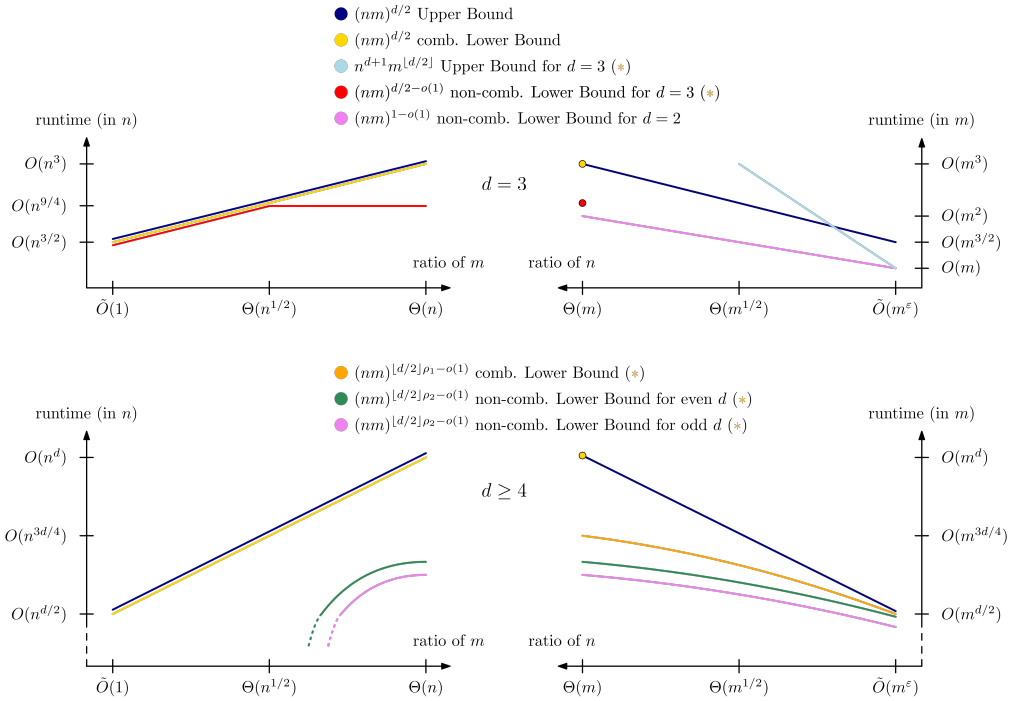


Figure 1 Schematic figure of upper and lower bounds for directed Hausdorff under Translation in 3-D (upper figure) and d-D for $d \geq 4$ (lower figure), novel results are marked with $(*)$. ρ_1, ρ_2 are functions on the ratio of n, m which correspond to values described in Theorem 3.7.

under translation transfers to faster approximation algorithms for the continuous version. Furthermore, the discrete version recently received interest from the perspective of a structural fine-grained complexity theory: the directed discrete Hausdorff distance under translation is part of a problem pair that is fine-grained complete for the class $\mathsf{FOP}_{\mathbb{Z}}$, a class of polynomial-time problems formed from first-order properties on finite additive structures [14] (further discussion can be found in the full version).

Previous works on Hausdorff distances under translation mostly focus –with notable exceptions such as [7, 9]– on a single particular variant, often considering particular dimensions d and the *balanced* case, in which all inputs sets (i.e., P, Q, and in the discrete case also T) have roughly equal size n. In this work, we aim to give a more comprehensive analysis, which reveals an intricate interplay between the dimensionality, symmetry (directed/undirected and the relationship between n and m), and discreteness.

1.1 Our Results

Our first question concerns the fine-grained complexity of the algorithmically most studied variant, the (continuous) directed Hausdorff distance under translation. The currently best upper bounds are $O(nm \log nm)$ for $d \leq 2$ [11], $O((nm)^{d/2} \log nm)$ for $d = 3$ [10][2], and $O((nm)^{d/2}(\log\log nm)^{O(1)})$ for general $d \geq 4$ [9]. This suggests the following question:

Question 1: *Is $(nm)^{d/2 \pm o(1)}$ the time complexity of directed Hausdorff L_∞-distance under Translation?*

[2] While [10] only considers the case $m = n$, their analysis also holds for distinct n, m. A short explanation of their approach can also be found in the full version.

Table 1 An overview of known results of the L_∞-Hausdorff distance under Translation in 1-D and 2-D for the balanced case, i.e., $n = m$ ($= t$ in the discrete setting). Upper Bounds are marked in blue, lower bounds in green, while results under non-standard hypothesis are additionally marked with (†). Novel results are denoted with (∗) . Results referring to the full version are marked with §.

$d = 1$	**Directed**	**Undirected**
Continuous	$\tilde{O}(n^2)$ ([11]) (∗) $n^{2-o(1)}$ (§)†	$\tilde{O}(n)$ ([20]) $\Omega(n)$
Discrete	$\tilde{O}(n^2)$ (§) (∗) $n^{2-o(1)}$ (3.10)†	$\tilde{O}(n)$ (§) $\Omega(n)$

$d = 2$	**Directed**	**Undirected**
Continuous	$\tilde{O}(n^2)$ ([11]) $n^{2-o(1)}$ ([7])	$\tilde{O}(n^2)$ ([11], 2.3) $n^{2-o(1)}$ ([7])
Discrete	$\tilde{O}(n^2)$ (§) (∗) $n^{2-o(1)}$ (3.14)†	$\tilde{O}(n^2)$ (§) (∗) $n^{2-o(1)}$ (3.14)†

A conditional lower bound of Bringmann and Nusser [7] (based on the Orthogonal Vectors Hypothesis) answers this question positively for $d = 2$. Subsequently, Chan [9] gave evidence of optimality even for $d \geq 3$, by proving a conditional lower bound of $(nm)^{d/2-o(1)}$ for *combinatorial* algorithms[3] in the case that $m \leq n$. However, proving such a conditional lower bound against *general* algorithms for $d \geq 3$ (or better conditional lower bounds than $(nm)^{1-o(1)}$ for the case $n \ll m$) remained open until this work. Furthermore, for $d = 1$, no $\tilde{O}(n + m + (nm)^{d/2}) = \tilde{O}(n + m)$ algorithms are known, raising the question whether any superlinear lower bound can be proven for $d = 1$.

As our first main contribution, we give a perhaps surprising *negative* answer to Question 1: For $d = 3$ and $n = m^{o(1)}$, there exists an almost-linear time $m^{1+o(1)}$ algorithm, showing that in an unbalanced setting, the upper bound of $\tilde{O}((nm)^{d/2})$ can indeed be beaten (Section 3.4). Interestingly, this stands in contrast to the case of moderately small m: For $d = 3$ and any $m \leq \sqrt{n}$ we prove a tight conditional lower bound of $(nm)^{d/2-o(1)}$, by combining the proof techniques of Chan's combinatorial k-clique lower bound [9] and the tight 3-uniform hyperclique lower bound for Klee's measure problem in 3D [19] (Section 3.2). This reveals that the time complexity of the directed Hausdorff distance under translation is *inherently asymmetric* in n and m and breaks with tradition for algorithmic results on the Hausdorff distance under translation.

We investigate the case of small n further and observe that our algorithmic approach could plausibly generalize to an $\tilde{O}(n^{d+1}m^{\lfloor d/2 \rfloor})$ algorithm (we elaborate on the missing ingredient in Section 3.4) – such an algorithm would beat the baseline algorithm for all odd d and small values of n. We also show that such an algorithm would indeed be conditionally tight: we also prove a corresponding lower bound of $m^{\lfloor d/2 \rfloor - o(1)}$ for all $d \geq 4$, assuming the 3-uniform hyperclique hypothesis (Section 3.1).

Our conditional lower bounds can be adapted to the case of $n = \Theta(m)$, but no longer match the upper bound. In fact, we obtain a conditional lower bound of $n^{2.25-o(1)}$ for $d = 3$ which still leaves a gap of $n^{0.75 \pm o(1)}$ to the upper bound of $\tilde{O}(n^3)$. It remains an open problem to settle the full time complexity for *any* relationship between n and m. An intriguing possibility consistent with our results is the existence of an $\tilde{O}(n^{\lceil d/2 \rceil}m^{\lfloor d/2 \rfloor})$-time algorithm.

[3] Here, combinatorial algorithms refers to algorithms that avoid the use of algebraic techniques underlying fast matrix multiplication algorithms beating $n^{3-o(1)}$ running time.

We turn to our second main question.

Question 2: *How does the choice of dimension, directed vs. undirected and discrete vs. continuous affect the time complexity?*

We observe that the time complexity of computing Hausdorff distances under translation exhibits an intricate interplay between the different aspects of the problem, illustrated in Table 1 and Figure 1.

On directed vs. undirected

A folklore reduction from computing the undirected Hausdorff distance to computing the directed Hausdorff distance under translation (Observation 2.3) shows that the undirected case is never harder than the (balanced) directed case. Could there exist a reduction in the other direction?

We show that in full generality, such a reduction appears unlikely: While for all $d \geq 3$, we can transfer our conditional lower bounds for the directed case to analogous bounds in the undirected case (analogous to the OVH-based lower bound for $d = 2$ [7], which also transfers to the undirected case) (Section 3.5), we obtain a conditional separation for $d = 1$. Notably, Rote [20] obtains a $\tilde{O}(n+m)$-time algorithm for computing the undirected Hausdorff distance under translation in $d = 1$. For the directed setting, we give a fine-grained reduction from the quadratic-time L_∞-Necklace Alignment problem [6], which in turn is at least as hard as the MaxConv LowerBound problem studied in [12]. We remark that we are not aware of any conditional lower bound for MaxConv LowerBound under an established hypothesis from fine-grained complexity theory; however, the question of subquadratic equivalence between this problem and the well-studied $(\min, +)$-convolution problem was raised as an open problem in 2017 [12]. We believe that our reduction to the (discrete or continuous) directed Hausdorff distance adds further reason to study the apparent quadratic-time complexity of MaxConv LowerBound, in hopes to either refute or justify this hardness barrier.

Circumventing the separation in $d = 1$, we give a general reduction from the undirected setting to the directed setting that blows up the dimension (Section 3.5). This allows us to transfer the lower bounds from the directed case for $d = 1$ to the undirected case in $d = 2$. In particular, we conclude that if MaxConv LowerBound truly requires quadratic-time, the undirected setting is near-linear time solvable if and only if $d = 1$.

On discrete vs. continuous

The tight lower bound based on OVH [7] convincingly settles the fine-grained complexity of the continuous setting in $d = 2$. In contrast, the only justification of hardness of the discrete setting is given by our reductions from MaxConv LowerBound and L_∞-Necklace Alignment (see above and the full version). Is there any reason why a tight lower bound based on OVH does not exist for the discrete setting?

We find evidence for the non-existence of such a reduction by showing a fine-grained reduction from (directed or undirected) discrete Hausdorff distance under translation in $d \leq 3$ to the All-Ints 3SUM problem (in the full version). In particular, this implies a tight fine-grained reduction from the balanced setting to 3SUM. Thus, establishing quadratic-time OVH-hardness of the balanced, discrete setting implies a tight reduction from OV to 3SUM, which is a central open problem in fine-grained complexity theory, and suffers from barriers based on the Nondeterministic Strong Exponential Time Hypothesis, see [8].

We remark that we obtain further connections from the class of additive problem $\mathsf{FOP}_{\mathbb{Z}}$ to the discrete variants. For space reasons, these results are deferred to the full version.

2 Preliminaries

A brief definition of relevant problems as well as the used hardness hypotheses can be found in the full version. We there also give a brief overview of known algorithms for the continuous and discrete variant.

We recall the Hausdorff distance as a distance measure between two point sets P, Q of sizes n, m. The metric space used for this paper will be $(\mathbb{R}^d, L_\infty)$. The undirected (symmetric) and directed (asymmetric) variant are defined as:

$$\delta_{\overrightarrow{H}}(P,Q) \coloneqq \max_{p\in P}\min_{q\in Q}\|p-q\|_\infty \qquad \delta_H(P,Q) \coloneqq \max\{\delta_{\overrightarrow{H}}(P,Q), \delta_{\overrightarrow{H}}(Q,P)\}$$

We show almost all of our conditional lower bounds already for the *decision variant* of Hausdorff under Translation, which receives a test distance $\delta > 0$ as additional input[4]. Clearly, the corresponding optimization problems, finding a translation minimizing δ, are at least as hard. Specifically, we study:

▶ **Definition 2.1** (Hausdorff under Translation (*HuT*)). *Given sets $P, Q \subset \mathbb{R}^d$ of n and m points, respectively, and a real value $\delta > 0$. Decide whether:*

$$\exists \tau \in \mathbb{R}^d : \delta_{\overrightarrow{H}}(P+\tau, Q) \le \delta \quad \equiv \quad \exists \tau \in \mathbb{R}^d\ \forall p \in P\ \exists q \in Q : \|(p+\tau)-q\|_\infty \le \delta$$

▶ **Definition 2.2** (Discrete Hausdorff under Translation (*DiscHuT*)). *Given sets $P, Q \subset \mathbb{Z}^d$ of n and m points, respectively, a real value $\delta > 0$ as well as a set of at most t translations $T \subset \mathbb{Z}^d$. Decide whether:*

$$\exists \tau \in T : \delta_{\overrightarrow{H}}(P+\tau, Q) \le \delta \quad \equiv \quad \exists \tau \in T\ \forall p \in P\ \exists q \in Q : \|(p+\tau)-q\|_\infty \le \delta$$

Note that our definition of the discrete variant assumes all input entries to be polynomially bounded integers[5].

To obtain the *undirected* versions of the above problems, we replace $\delta_{\overrightarrow{H}}(P+\tau, Q)$ by $\delta_H(P+\tau, Q)$, and denote the problem by *(Disc)uHuT*. If not specified otherwise, we consider the directed versions. We call the case in which all input sets have roughly equal size, i.e., $Q = \Theta(|P|)$ and additionally for the discrete case $|T| = \Theta(|P|)$, the *balanced* parameter setting, otherwise we say the parameters are *lopsided*.

Lastly, we give a folklore reduction from undirected to directed Hausdorff under Translation that works for both the continuous and discrete variant. By setting $\sigma = (M, 0, \ldots, 0)$ for a sufficiently large M, we see that $\delta_H(P+\tau, Q) = \delta_{\overrightarrow{H}}((P \cup (\sigma - Q)) + \tau, Q \cup (\sigma - P))$.

▶ **Observation 2.3.** *The undirected (Discrete) Hausdorff under Translation problem with input sizes n, m reduces to the directed version with input sizes $n' = m' = n + m$.*

While this reduction introduces additional points to the input, we discuss in the full version that an input size maintaining reduction to a single instance of directed HuT is unlikely.

As our model of computation, we use the Real RAM for continuous HuT, while for the discrete HuT variant with point sets in $\mathbb{Z}^d$ we assume a Word RAM machine model with word size $\Theta(\log n)$.

[4] Note that the case $\delta = 0$, i.e., finding an exact match, allows an almost-linear-time algorithm [13] and is thus uninteresting for our purposes.

[5] One may also assume rational coordinates, which is equivalent by a scaling argument.

2.1 Translation Problems

For clarity of presentation, it is often useful to view the Hausdorff under Translation decision problem as a translation problem involving certain *shapes*.

▶ **Definition 2.4** (Translation Problem). *Given a set of n points P and m shapes $\mathcal{Q}$ in $\mathbb{R}^d$. Compute a translation $\tau \in T \subseteq \mathbb{R}^d$ such that*

$$\forall p \in P \ \exists Q \in \mathcal{Q} : p + \tau \in Q.$$

For an instance of HuT with point sets P, Q and $\delta > 0$, if we construct hypercubes $\mathcal{Q}$ with side length 2δ and centers $q \in Q$, the region $T^* = \bigcap_{p \in P} \bigcup_{\hat{Q} \in \mathcal{Q}} \hat{Q} - p$ is exactly the region of feasible translations, i.e., we have $\tau \in T^* \iff \forall p \in P \ \exists q \in Q : \delta_{\overrightarrow{H}}(P + \tau, Q) \leq \delta$ [1]. We call this case, where all shapes $\mathcal{Q}$ are hypercubes of the same size, the *Translation Problem with Hypercubes (TPwC)*. In case all shapes of Q are axis-aligned boxes (of arbitrary size), we call it the *Translation Problem with Boxes (TPwB)*. While the TPwC problem is equivalent to HuT, there is no reduction to HuT known within the same ambient space from the Translation Problem with Boxes or even from the translation problem with hypercubes of different sizes. We further use a third variant.

▶ **Definition 2.5** (Translation Problem with Orthants (TPwO)). *Given a target box B with a maximum side length δ_0. Given a set of k Translation Problem with Hypercube instances, each on a point set P_i and hypercubes $\mathcal{Q}_i$ with side length $\delta > \delta_0$, such that $\sum_{i \in [k]} |P_i| = n$, $\sum_{i \in [k]} |\mathcal{Q}_i| = m$. Compute a translation $\tau \in \mathbb{R}^d$ such that $\forall i \in [k] : P_i + \tau \subset B$ and*

$$\forall i \in [k] \ \forall p_i \in P_i \ \exists Q_i \in \mathcal{Q}_i : p_i + \tau \in Q_i.$$

We call the shapes of the TPwO instance, i.e., the shapes of its sub-instances, orthants as within the target box all hypercubes are indistinguishable from orthants. An important property of this problem, which we will use later, is that we may fix the side-length of all orthants to a value $\delta > \delta_0$. Similar to [9, Lemma 3], the TPwO problem reduces to the TPwC and thus has a reduction to HuT, its proof is deferred to the full version. A converse reduction, from HuT to TPwO, is generally not known.

▶ **Lemma 2.6.** *The Translation Problem with Orthants on n points and m orthants reduces in linear time to the Translation Problem with Hypercubes on $\Theta(n)$ points and $\Theta(m)$ hypercubes.*

3 Technical Overview for Selected Results

In this section, we discuss the main ideas of our lower bounds and algorithm. We present the general structure of the proofs and give sketches on the most important technical details. For space reasons, we defer further details as well as a complete discussion of our results and further extensions to the full version.

3.1 (Non-)Combinatorial Lower Bounds for Arbitrary n, m

For our lower bounds in higher dimensions ($d \geq 4$), we give a reduction in three steps from the k-hyperclique problem to HuT, see the full version for a formal definition and its hardness hypothesis.

In the first step, we encode any k-tuple of vertices of a hypergraph $H = (V, E)$ as cells inside a $\lfloor d/2 \rfloor$-dimensional hypercube, see Figure 2. The used encoding was introduced by Chan [9]. For any non-edge $\bar{e} \in \binom{V}{3} \setminus E$, the *feasible region* of $\bar{e}$ are the cells inside the

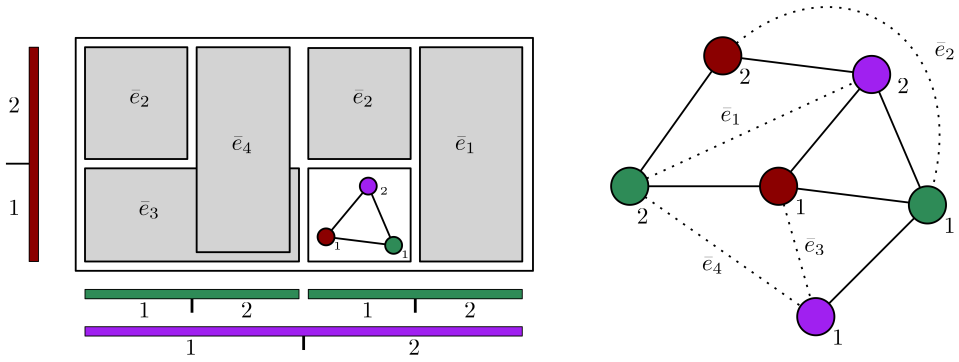


Figure 2 An example of an encoding of a 3-partite graph to a 2-D box (skewed cube). Normal lines represent edges, dotted lines represent a subset of non-edges. The gray boxes correspond to the forbidden regions of the respective non-edge. Note that in our reductions, we aim to cover the complement of these regions, the feasible regions.

hypercube for which the corresponding k-tuple does not include $\bar{e}$. In contrast, the cells for which the corresponding k-tuple includes all vertices of $\bar{e}$ are called the *forbidden region*. For each non-edge, we construct a covering of its feasible region with axis-aligned boxes[6], see Lemma 3.5 for the exact covering. As a k-clique is a k-tuple which does not contain any non-edge, a cell that lies within the feasible regions of all non-edges encodes a k-clique. In Lemma 3.6 we reduce finding a cell inside all feasible regions to the intermediate Translated Box-Shape Problem (Definition 3.2) with our desired values of n, m.

In the second step, the resulting Translated Shape Problem is reduced to the Translation Problem with Boxes, see Lemma 3.3.

Lastly, in our third step we reduce the Translation Problem with Boxes in $\lfloor d/2 \rfloor$-D to the Translation Problem with Orthants in d dimensions with n points and m orthants, see Lemma 3.1. As given in Lemma 2.6, the Translation Problem with Orthants has a linear time reduction to directed Hausdorff under Translation, which yields our lower bound.

We go through the steps in backwards order, starting with the last, third step.

Step 3: From Boxes to Orthants

We reduce the Translation Problem with Boxes to the Translation Problem with Orthants. We implement the natural idea to cover boxes of arbitrary aspect ratios by orthants (large hypercubes), at the cost of doubling the underlying dimension. As this basic idea allows additional, unintended solutions, we need a technical translation gadget that ensures soundness. Figure 3 gives a visual intuition of the proof in 1-D.

▶ **Lemma 3.1.** *The Translation Problem with Boxes in d-D with input parameters n, m reduces in linear time to the Translation Problem with Orthants in $2d$-D with input parameters $n' = \Theta(n), m' = \Theta(m)$.*

Proof Sketch. We define the coordinate-wise transformations

$$c : p \mapsto (p, p), \qquad c^+ : [a, b] \mapsto [a, \infty) \times (-\infty, b], \qquad c^- : [a, b] \mapsto (-\infty, b] \times [a, \infty).$$

[6] As we only consider axis-aligned boxes in this paper, we may only call these boxes.

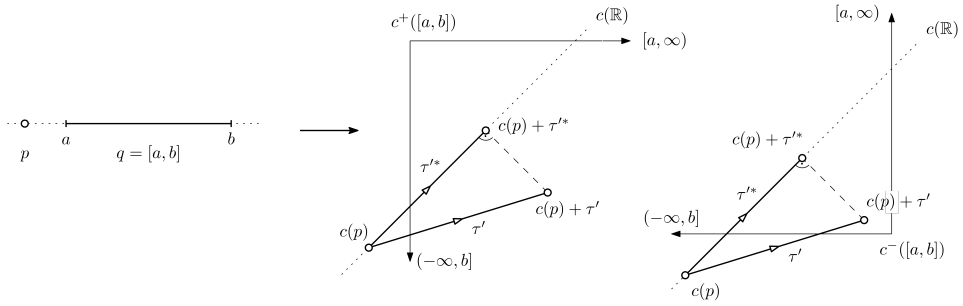


Figure 3 The reduction of a box in 1-D, i.e., an interval $[a, b]$, to orthants in 2-D. We prove that for each feasible translation τ' there exists a feasible translation τ'^* which lies on the diagonal $c(\mathbb{R})$. For that, we always assume that the translation τ' lies between the diagonal $c(\mathbb{R})$ and the vertex of the orthant, i.e., for this example the orthant of c^- is relevant.

For points in $\mathbb{R}^d$, we define the transformation f as the coordinate-wise application of c. For our analysis, we further apply f to sets or intervals in which case we apply f element-wise. For boxes in $\mathbb{R}^d$, we define the transformations $f_1, ..., f_{2^d}$ as all possible combinations of the coordinate-wise application of c^+, c^-. Note that the images of f_i are orthants.

Given an instance of the Translation Problem with Boxes with P being the point set and $\mathcal{B}$ being the set of boxes. We construct the sub-instances $(f(P), f_1(\mathcal{B})), ..., (f(P), f_{2^d}(\mathcal{B}))$ of the resulting Translation Problem with Orthants instance, the target box is simply the bounding box of $\mathcal{B}$. We claim that

$$\begin{aligned} &\exists \tau \in \mathbb{R}^d \ \forall p \in P \ \exists B \in \mathcal{B} : p + \tau \in B \\ \iff &\exists \tau' \in \mathbb{R}^{2d} \ \forall i \in [2^d] \ \forall p' \in f(P) \ \exists C \in f_i(\mathcal{B}) : p' + \tau' \in C. \end{aligned}$$

For the proof, we see that a feasible translation $f(\tau) \in f(\mathbb{R}^d)$ for the right side for *any* $i \in [2^d]$ is a feasible translation for the left side, so we prove that a feasible translation $\tau' \notin f(\mathbb{R}^d)$ can be transformed to a feasible translation $\tau'^* \in f(\mathbb{R})$. Choosing the $i \in [2^d]$ such that τ' always lies on the correct side of $f(\mathbb{R}^d)$, where all vertices of our constructed orthants $f_i(\mathcal{B})$ lie, enables us to move τ' to its closest point τ'^* on $f(\mathbb{R}^d)$, giving $p' + \tau' \in C \implies p' + \tau'^* \in C$ for all $p' \in f(P), C \in f_I(\mathcal{B})$. The full proof details can be found in the full version. ◀

Step 2: The intermediate Translated Box-Shape Problem

We introduce an intermediate problem, the Translated Box-Shape Problem, similar to the one of [9, Problem 4] using hypercubes. In contrast to the Translation Problem with Boxes, it allows composing boxes into shapes for which translated copies can be instantiated with a single translation object. Thus, a set of n^2 boxes, structured appropriately, can be described by a shape of n boxes and n translation objects.

▶ **Definition 3.2.** (Translated (Box-)Shape Problem). *Let $\mathcal{Z}$ be a set of shapes in $\mathbb{R}^d$, composed of a union of boxes. Let m be the total number of boxes over all shapes of $\mathcal{Z}$. Given a set $\mathcal{S}$ of n objects where each object is the translate of some shape in $\mathcal{Z}$, decide whether $\bigcap_{S \in \mathcal{S}} S \neq \emptyset$.*

We further give a reduction from the Translated Shape Problem to the Translation Problem with Boxes, analog to [9, Lemma 3]. Its proof again is deferred to the full version.

▶ **Lemma 3.3.** *The Translated (Box-)Shape Problem with n translation objects and a total of m boxes reduces to the Translation Problem with Boxes with $|P| = \Theta(n)$ and $|\mathcal{Q}| = \Theta(m)$.*

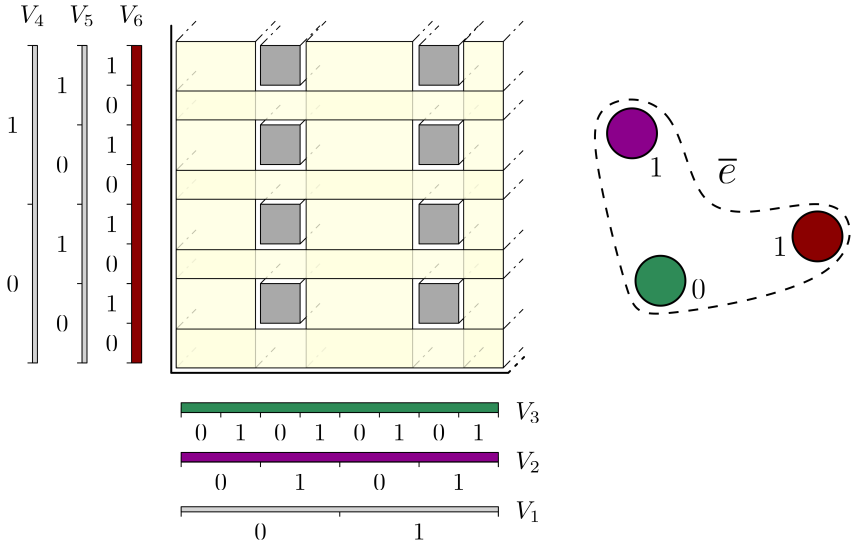


Figure 4 An example of the feasible region of a non-edge $\bar{e} \in V_2 \times V_3 \times V_6 \setminus E$ covered by boxes, depicted in yellow, within a d-dimensional hypercube projected to 2 dimensions.

Lastly, we give a simple observation allowing, by repeated application, to compose arbitrary many individual Translated (Box-)Shape Problems without additional overhead.

▶ **Observation 3.4.** *Given two instances of the Translated Shape Problem with inputs $\mathcal{S}_1, \mathcal{Z}_1$ and $\mathcal{S}_2, \mathcal{Z}_2$ of size n_1, m_1 and n_2, m_2 in a shared universe. Deciding whether $\bigcap_{S \in \mathcal{S}_1 \cup \mathcal{S}_2} S \neq \emptyset$ reduces to a single instance of the Translated Shape Problem with $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$, $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2$ of size $n \leq n_1 + n_2$ and $m \leq m_1 + m_2$.*

Step 1: From k-Hyperclique to the Translated Box-Shape Problem

In the remaining step, along the basic idea presented in the beginning of Section 3.1, we encode the k-hyperclique problem to a d-dimensional hypercube and reduce finding a cell that lies in the feasible regions of all non-edges to the Translated (Box-)Shape Problem with the desired distribution of parameters n, m. As our approach will also provide novel combinatorial lower bounds, we more generally consider a u-uniform hypergraph with $u \in \{2, 3\}$.

We first give the encoding of the hypergraph to a d-dimensional hypercube and show that the feasible regions of the non-edges can be covered efficiently. The same encoding for a graph can exemplarily be seen in Figure 2, note that for illustrative reasons we there have not covered the feasible but the forbidden regions.

▶ **Lemma 3.5.** *Let $d \in \mathbb{N}$ and $u \in \{2, 3\}$ be constant and let $k \in \mathbb{N}$ be constant with $d|k$. Given a u-uniform hypergraph $H = (V, E)$ with $|V| = n$, where V is partitioned into k color classes $V = V_1 \cup V_2 \cup ... \cup V_k$. There is an encoding of $V_1 \times ... \times V_k$ to a d-dimensional hypercube so that for each $\bar{e} \in \binom{V}{u} \setminus E$ the feasible region of $\bar{e}$ can be covered by $O(n^{k/d-1})$ boxes, which can be computed in $O(n^{k/d-1})$ time.*

Proof Sketch. By scaling, we assume that the d-dimensional hypercube has a side-length of $n^{k/d}$. Given the numbers $x_1, ..., x_{k/d} \in [n]$, we use the canonical function

$$\text{ind} : (x_1, ..., x_{k/d}) \mapsto x_1 \cdot n^{k/d-1} + x_2 \cdot n^{k/d-2} + ... + x_{k/d}$$

to address an integer point in the range $[0, n^{k/d} - 1]$ We denote each unit hypercube as a cell, its relative origin can be addressed by $(\text{ind}(X_1), ..., \text{ind}(X_d))$ where $X_1, ..., X_d \in [n]^{k/d}$.

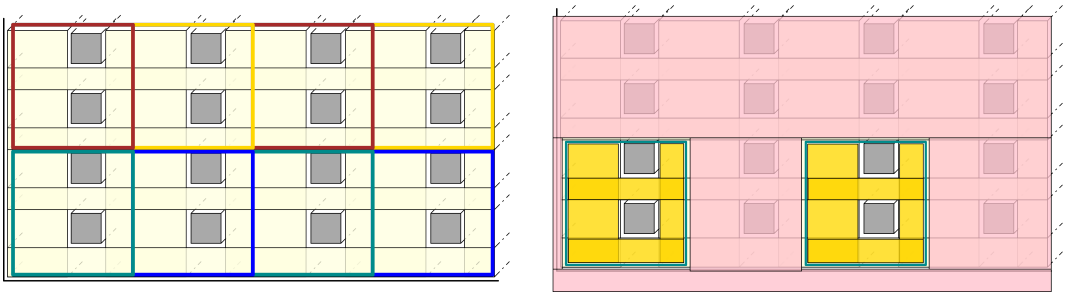


Figure 5 Left: The decomposition of a (skewed) hypercube in 2-D into four identical sub-regions, marked in red, yellow, green, and blue. Boxes in equal color denote a common sub-region, here in each row there are two sub-regions that are split into two parts each.
Right: The boxes used to cover the feasible region of the green sub-regions, in yellow, as well as the boxes covering the complement of the green sub-region, marked in pink.

The encoding of H to the hypercube is the same as used in [9]. We evenly assign the k color-classes to the d dimensions, i.e., the color classes $V_1, ..., V_{k/d}$ are assigned to the first dimension, the classes $V_{k/d+1}, ..., V_{2k/d}$ to the second dimension, etc. A k-tuple $(v_1, ..., v_k) \in V_1 \times ... \times V_k$ is bijectively assigned to the cell

$$\text{cell}((v_1, ..., v_k)) = (\text{ind}(v_1, ..., v_{k/d}), \text{ind}(v_{k/d+1}, ..., v_{2k/d}), ..., \text{ind}(v_{(d-1)k/d}, ..., v_k)).$$

For any non-edge $\bar{e} \in V^u \setminus E$, we cover the feasible region of $\bar{e}$ using at most $O(dn^{k/d-1})$ boxes $\mathcal{B}$, as can be seen in Figure 4. A full proof can be found at in the full version. ◀

Using Lemma 3.5, we can already turn a k-(hyper-)clique instance into a Translated (Box-)Shape instance with $n = O(|V|^u), m = O(|V|^{k/d-1+u})$ as we can treat each feasible region of the $O(|V^u|)$ non-edges as an independent shape with $O(|V|^{k/d-1})$ boxes each. Observation 3.4 allows to compose the resulting independent instances.

To extend the reduction to arbitrary parameter choices of n, m, independently for each non-edge we decompose the hypercube into n smaller identical parts such that in each part the feasible region of a non-edge can be covered by m boxes as shown in Lemma 3.5. This allows a reduction to the Translated Shape Problem with m boxes and n translation objects.

▶ **Lemma 3.6.** *Let $\lambda \in [0, 1]$, $u \in \{2, 3\}$ be constant, and $d, k \in \mathbb{N}$ be constants with $d|k$. Given a u-regular hypergraph $H = (V, E)$ and its encoding to a d-dimensional hypercube as constructed in Lemma 3.5. We reduce the problem of finding a k-(hyper-)clique in H in $O(|V|^{k/d})$ time to an instance of the Translated (Box-)Shape Problem with $n = O(|V|^{u\lambda k/d+u})$ translates of common shapes consisting of $m = O(|V|^{(1-\lambda)k/d+u})$ boxes.*

Proof Sketch. By Observation 3.4, it suffices to consider the feasible regions of a single non-edge $\bar{e}$ and reduce finding a cell that lies in its feasible regions to a Translated (Box-)Shape instance with $n' = O(|V|^{u\lambda k/d})$ translates of a single shape with $m' = O(|V|^{(1-\lambda)k/d})$ boxes.

We consider the encoding of the hypergraph to the d-dimensional hypercube as given in Lemma 3.5. As $\bar{e}$ is a u-tuple, its included vertices are only encoded in up to u dimensions, it thus suffices to consider the hypercube projected to these u dimensions.

We construct $((|V|^{k/d})^\lambda)^u = n'$ many sub-regions that decompose the u-dimensional hypercube: We decompose each single dimension into $(|V|^{k/d})^\lambda$ many (not necessarily continuous) slices that are identical under translation, i.e., when translated so that two slices have the same relative origin, the feasible and forbidden regions of $\bar{e}$ within these slices match. These one-dimensional slices can be combined independently in all dimensions, yielding the desired decomposition of the full u-dimensional hypercube into identical sub-regions.

The feasible region of $\bar{e}$ in each sub-region can be covered by m boxes. As the sub-regions are identical, we can construct a Translated (Box-)Shape instance with n' translation objects and a single shape with m boxes. A full proof can be found in the full version. ◀

Resulting Lower Bounds

Combining our results from Lemmas 3.1, 3.3, and 3.6, we get lower bounds for arbitrary distributions of n, m. However, as the bounds deteriorate with growing values of n, we only get tight lower bounds for $n = m^{o(1)}$. We consider other cases in the full theorem statement which can be found with its proof in the full version.

▶ **Theorem 3.7.** *Let $\varepsilon > 0, \lambda \in (0, 1], d \in \mathbb{N}_{\geq 4}$ be constant. Under the k-hyperclique hypothesis, there is no $O((nm)^{\lfloor \frac{d}{2} \rfloor \frac{\lambda+3}{3\lambda+3} - \varepsilon})$ algorithm for directed HuT in d-D with $n = \Theta(m^\lambda)$.*

3.2 Tight Non-Combinatorial Lower Bound in 3-D

Chan [9] has proven a tight lower bound that applies for distributions $m \in O(n)$ in 3-D. We give a non-combinatorial version of this $(nm)^{d/2-o(1)}$ lower bound for $m \in O(n^{1/2})$.

We reduce from the k-Hyperclique problem, combining Chan's ideas with the work of Künnemann [19] using *prefix covering designs*. As done in Section 3.1, we encode a k-tuple of vertices inside a d-dimensional hypercube. However, we use a redundant encoding based on the prefix covering designs, that is, each color class is encoded more than once to our hypercube. The full proof is presented in the full version.

▶ **Corollary 3.8.** *Let $\varepsilon > 0$. For any $\lambda \in (0, 1/2]$, there is no $O((nm)^{3/2-\varepsilon})$ (non-combinatorial) algorithm for directed Hausdorff under Translation in 3-D with $m = \Theta(n^\lambda)$ under the k-hyperclique hypothesis.*

3.3 Connections to Additive Problems

We draw several connections between the Hausdorff distance under Translation and several additive problems. Below we highlight a conditional lower bound in 1D and conditional upper bound in 3D. Omitted proofs and additional results are given in the full version.

We first show that even for one-dimensional instances, directed HuT inherits hardness from quadratic-time barriers from problems such as *MaxConv LowerBound*.

▶ **Definition 3.9** (MaxConv LowerBound, MaxConvLB)**.** *Given integer arrays A, B, C of length[7] n. Determine whether $C[k] \leq \max_{i+j=k}(A[i] + B[j])$ holds for all $k \in \{2, ..., n\}$.*

In an investigation of hardness results from $(\min, +)$-convolution, Cygan, Mucha, Wegrzycki and Wlodarczyk [12] studied the closely related MaxConv LowerBound problem and gave a subquadratic reduction to L_∞-Necklace Alignment (in turn, L_∞-Necklace Alignment admits a subquadratic reduction to $(\min, +)$-Convolution [6]). Cygan et al. raised the natural question whether MaxConv LowerBound is subquadratic equivalent to $(\min, +)$-convolution; to the best of our knowledge, neither a positive or negative resolution is known.

For the (continuous) directed HuT we get hardness by a reduction from the Linear Alignment problem, shown in the full version. Further, using a reduction from L_∞-Necklace Alignment to Linear Alignment, we get a reduction from MaxConv LowerBound to directed

[7] Other common definitions use for C an array of size $2n$. For convenience, we stick to this definition.

HuT. Interestingly, we obtain such a lower bound even for discrete directed HuT by directly reducing from MaxConv LowerBound. These reductions illuminate why no subquadratic algorithms are known for directed variants, despite near-linear algorithms for their undirected counterparts [20].

▶ **Theorem 3.10.** *If 1-dimensional discrete HuT can be solved in time $T(n)$, then the MaxConv LowerBound problem can be solved in time $O(T(n))$.*

Proof sketch. Given arrays A, B, C of positive values as inputs from MaxConv LowerBound. We construct T, P, Q as arrays for DiscHuT. By abuse of notation, we allow Q to be indexed negatively. We define $T = [-(kM + C[k])]_{k\in[n]}$, $P = [iM + A[i] + M/4]_{i\in[n]}$, $Q = [-zM + M/4]_{z\in\{-n,\dots,0\}} \cup [-(jM + B[j])]_{j\in[n]}$, $\delta = \frac{M}{4} - 1$, where M is a large number with $M > 4 \cdot \max_{i,j,k} (A[i] + B[j] + C[k])$.

Now, if A, B, C is a NO-instance to MaxConv LowerBound, then there is a k^* such that for all $i < k^*$ we have $A[i] + B[k^* - i] < C[k^*]$. From this we can derive that $|p_i + \tau_{k^*} - q_{k^*-i}| \leq \delta$, where p_i, q_{k^*-i}, and τ_{k^*} denote the points in P, Q, and T, respectively, corresponding to $A[i]$, $B[k^* - i]$ and $C[k^*]$. Thus, we have a YES-instance to DiscHuT. Analogously, a YES-instance to MaxConv results in a NO-instance to DiscHuT. See the full version for the full proof. ◀

We give an upper bound to discrete HuT, which stems from a reduction from 3-dimensional Discrete HuT to 3-SUM. On a very high level, the basic idea consists of a complement trick, which enables a quantifier switch in the discrete HuT problem.

▶ **Corollary 3.11.** *Let $d \leq 3$. If 3-SUM can be solved in time $T(n)$, then we can solve d-dimensional balanced Discrete HuT in time $\tilde{O}(T(n \log(n)) + n \log(n))$.*

3.4 A Faster Algorithm for $m \gg n$

For $n = m^{o(1)}$, Theorem 3.7 gives a $(nm)^{\lfloor d/2 \rfloor - o(1)}$ lower bound, which for even d matches the best known algorithm running in time $O((nm)^{d/2})$ [9], while for odd d it seems to have some slack. Intuitively, one might expect that the known $(nm)^{d/2-o(1)}$ combinatorial lower bound for $m = O(n)$ [9] also transfers to the case of $m \gg n$. Surprisingly, in 3-D the $(nm)^{1-o(1)}$ lower bound of [7] is tight for $n = m^{o(1)}$ as we give an $O(m^{1+o(1)})$ algorithm.

The algorithm uses a structural insight from [10, Theorem 7], which we turn algorithmic. The full description and proof can be found in the full version. Our algorithm relies on an algorithm recently introduced by Agarwal and Steiger [2], which in 3-D computes the vertices of the union of m unit cubes in an output-sensitive runtime of $\tilde{O}(m)$. Assuming an algorithm that computes the vertices of the union of m unit hypercubes in d-D in time $\tilde{O}(m^{\lfloor d/2 \rfloor})$, our algorithm directly generalizes to higher dimensions with a runtime of $\tilde{O}(m^{\lfloor d/2 \rfloor} n^{d+1})$.

▶ **Theorem 3.12.** *Directed Hausdorff under Translation in 3-D can be solved in time $\tilde{O}(n^4 m)$.*

Its existence forbids $(nm)^{d/2-o(1)}$ lower bounds for arbitrary distributions of n, m for $d = 3$, and we give compelling intuition for a similar result in odd $d > 3$.

▶ **Corollary 3.13.** *Let $d = 3$, $\kappa > 0$ and set $\lambda = \frac{\kappa}{d+2}$. For $n = O(m^\lambda)$, directed Hausdorff under Translation in d-D can be solved in time $O(m^{\lfloor d/2 \rfloor} n^{d+1}) < O(m^{\lfloor d/2 \rfloor + \kappa})$.*

This result points to a discrepancy in complexity, at least for $d = 3$ but potentially also for all odd d. While we have a $\tilde{O}(m)$ algorithm for $n = m^{o(1)}$, there is a $(nm)^{d/2-o(1)}$ tight (combinatorial) lower bound for $m \in O(n)$ [9]. It remains unknown where the cutoff point is located, i.e., for which values of n, m do we start to get faster than $(nm)^{d/2 \pm o(1)}$ algorithms?

3.5 Transferring Lower Bounds to the Undirected Setting

As of now, all here presented lower bounds only apply to the directed Hausdorff under Translation setting. Chan [9] already raised the question if these bounds transfer to the undirected setting, however there are no such lower bounds known so far for $d \geq 3$. Only for $d = 2$, lower bounds for the undirected setting are known [7], which match the bounds for the directed variant. For $d = 1$, we have an optimal $\tilde{O}(n + m)$ algorithm [20].

We partially resolve this question by providing a reduction from the Translation Problem with Orthants to undirected Hausdorff under Translation, maintaining all instance parameters.

▶ **Theorem 3.14.** *If we can decide the undirected Hausdorff under Translation problem in d-D with point sets of size n, m in time $T(n, m)$, we can decide the Translation Problem with Orthants in d-D with n points and m orthants in time $T(n, m) + O(n + m)$.*

Proof Sketch. We only give intuition on the reduction to a uHuT instance from a single TPwC sub-instance of our TPwO input. The full proof is given in the full version.

For an individual TPwC instance, given the point set P and the set of hypercubes $\mathcal{C}$ of side length δ. Note that the target box B of our TPwO instance implies that the translated point set $P + T$ and the set of feasible translations T is restricted, we may assume that $T \subseteq [-\delta/3, \delta/3]^d$ and $P + T \subseteq B \subseteq b + [-2\delta/3, 2\delta/3]^d$, where $b \in \mathbb{R}^d$ is some relative origin.

We strategically place additional points in the TPwC sub-instance to allow a reduction to uHuT. These points are located on the vertices and middle points of all faces of a $b + [-\delta, \delta]^d$ extended target box. As the translation and translated point set are restricted, any hypercube that intersects the target box $b + [-2\delta/3, 2\delta/3]^d$ also intersects the extended target box $b + [-\delta, \delta]^d$ when translated. Thus, we can place auxiliary points such that under all translations each hypercube C of our TPwC instance has an auxiliary point which lies inside C. Having this property, the reduction to uHuT is imminent. ◀

All lower bounds for HuT in $d \geq 3$ dimensions presented so far, as well as the combinatorial $(nm)^{d/2-o(1)}$ lower bound for $m \in O(n)$ of Chan [9], can be directly applied to undirected Hausdorff under Translation as they use the Translation Problem with Orthants as an intermediate problem. At the cost of doubling the dimension using Lemma 3.1, we generally reduce any instance of directed Hausdorff under Translation to the undirected setting.

▶ **Corollary 3.15** (Excerpt from full version)**.** *We get the following results:*

- *For arbitrary n, m, there is no $O((nm)^{d/2-\varepsilon})$ combinatorial algorithm for undirected Hausdorff under Translation in d-D under the k-clique hypothesis.*
- *For $\min(n, m) \in O(\max(n, m)^{1/2})$, there is no $O((nm)^{3/2-\varepsilon})$ algorithm for undirected Hausdorff under Translation in 3-D under the k-hyperclique hypothesis.*
- *If balanced (discrete) undirected Hausdorff under Translation in 2-D can be solved in time $T_2(n)$, then MaxConv LowerBound can be solved in time $O(T_2(n) + n)$.*

References

1 P. K. Agarwal, M. Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. *J. Algorithms*, 17(3):292–318, 1994. doi:10.1006/jagm.1994.1038.

2 Pankaj K. Agarwal and Alex Steiger. An output-sensitive algorithm for computing the union of cubes and fat boxes in 3D. *proc48th*, 198:10:1–10:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.10.

3 Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. North Holland / Elsevier, 2000. doi:10.1016/B978-044482537-7/50004-8.

4 Helmut Alt, Kurt Mehlhorn, Hubert Wagener, and Emo Welzl. Congruence, similarity, and symmetries of geometric objects. *Discret. Comput. Geom.*, 3:237–256, 1988. doi:10.1007/BF02187910.

5 Gill Barequet and Sariel Har-Peled. Polygon Containment and Translation in Hausdorff-Distance Between Segment Sets are 3SUM-Hard. *International Journal of Computational Geometry & Applications*, 2001. doi:10.1142/S0218195901000596.

6 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Pătraşcu, and Perouz Taslakian. Necklaces, convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014. doi:10.1007/S00453-012-9734-3.

7 Karl Bringmann and André Nusser. Translating Hausdorff is hard: fine-grained lower bounds for Hausdorff distance under translation. *J. Comput. Geom.*, 13(2):30–50, 2021. doi:10.20382/JOCG.V13I2A3.

8 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *7th*, pages 261–270, 2016. doi:10.1145/2840728.2840746.

9 Timothy M. Chan. Minimum L_∞ Hausdorff distance of point sets under translation: Generalizing Klee's measure problem. In *proc39th*, volume 258, pages 24:1–24:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.SOCG.2023.24.

10 L. P. Chew, D. Dor, A. Efrat, and K. Kedem. Geometric pattern matching in d-dimensional space. *Discrete Comput Geom*, 21(2):257–274, 1999. doi:10.1007/PL00009420.

11 L. Paul Chew and Klara Kedem. Getting around a lower bound for the minimum Hausdorff distance. *Computational Geometry*, 10(3):197–202, June 1998. doi:10.1016/S0925-7721(97)00032-1.

12 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min,+)-convolution. *ACM Transactions on Algorithms*, 15(1):14:1–14:25, 2019. doi:10.1145/3293465.

13 Nick Fischer. Deterministic sparse pattern matching via the Baur-Strassen theorem. In *proc35th*, pages 3333–3353. SIAM, 2024. doi:10.1137/1.9781611977912.119.

14 Geri Gokaj and Marvin Künnemann. Completeness theorems for k-SUM and geometric friends: Deciding fragments of linear integer arithmetic. In *proc16th*, volume 325 of *LIPIcs*, pages 55:1–55:25, 2025. ISSN: 1868-8969. doi:10.4230/LIPIcs.ITCS.2025.55.

15 Michael T. Goodrich, Joseph S. B. Mitchell, and Mark W. Orletsky. Approximate geometric pattern matching under rigid motions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(4):371–379, 1999. doi:10.1109/34.761267.

16 Daniel P. Huttenlocher, Klara Kedem, and Micha Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete & Computational Geometry*, 9:267–291, 1993. doi:10.1007/BF02189323.

17 Daniel P. Huttenlocher, Gregory A. Klanderman, and William Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9):850–863, 1993. doi:10.1109/34.232073.

18 Sanjiv Kapoor and Michiel Smid. New techniques for exact and approximate dynamic closest-point problems. *SIAM Journal on Computing*, 25(4):775, 1996.

19 Marvin Künnemann. A tight (non-combinatorial) conditional lower bound for Klee's measure problem in 3D. In *proc63rd*, pages 555–566, 2022. doi:10.1109/FOCS54457.2022.00059.

20 Günter Rote. Computing the minimum Hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, 38(3):123–127, 1991. doi:10.1016/0020-0190(91)90233-8.

21 William Rucklidge. Lower bounds for the complexity of the graph of the hausdorff distance as a function of transformation. *Discret. Comput. Geom.*, 16(2):135–153, 1996. doi:10.1007/BF02716804.

22 Carola Wenk. *Shape Matching in Higher Dimensions*. PhD thesis, Freie Universität Berlin, 2003. doi:10.17169/refubium-8310.

Cauchy's Surface Area Formula in the Funk Geometry

Sunil Arya
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong

David M. Mount
Department of Computer Science, University of Maryland, College Park, MD, USA

Abstract

Cauchy's surface area formula expresses the surface area of a convex body as the average area of its orthogonal projections over all directions. While this tool is fundamental in Euclidean geometry, with applications ranging from geometric tomography to approximation theory, extensions to non-Euclidean settings remain less explored. In this paper, we establish an analog of Cauchy's formula for the Funk geometry induced by a convex body K in $\mathbb{R}^d$, for the Holmes–Thompson surface area. The formula is based on central projections to boundary points of K. We show that when K is a convex polytope, the formula reduces to a weighted sum of contributions associated with the vertices of K. Finally, as a consequence of our analysis, we derive a generalization of Crofton's formula for surface areas in the Funk geometry. By viewing Euclidean, Minkowski, Hilbert, and hyperbolic geometries as limiting or special cases of the Funk setting, our results provide a unified framework for these classical surface area formulas.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Convexity, Cauchy's formula, Funk geometry, Hilbert geometry, Crofton's formula, Holmes–Thompson surface area

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.8

Related Version *Full Version*: https://arxiv.org/abs/2601.16468 [9]

Funding *Sunil Arya*: Research Grants Council of Hong Kong, China project number 16214721.

1 Introduction

Cauchy's surface area formula is a surprising and elegant result in Euclidean geometry. It states that the surface area of any convex body is a constant multiple of the average $(d-1)$-dimensional measure of its orthogonal projections, or "shadows", taken over all possible directions. To state the result formally in any dimension $d \geq 1$, let S^d denote the d-dimensional unit sphere, λ_d denote the standard d-dimensional Lebesgue measure, ω_d denote the volume of a d-dimensional Euclidean unit ball, and $(\cdot \mid \cdot)$ denote orthogonal projection. Cauchy's formula states that for any convex body K in $\mathbb{R}^d$, its surface area is

$$\frac{1}{\omega_{d-1}} \int_{S^{d-1}} \lambda_{d-1}\big(K \mid u^\perp\big)\, du, \tag{1}$$

where $u^\perp$ is the linear subspace orthogonal to u. The related *Crofton formula* allows us to express the surface area of a convex body K as the average number of lines that intersect K, assuming that the lines are sampled from an appropriate measure.

Cauchy's formula is computationally fundamental, as it underpins algorithms in applications where estimating global geometric properties from lower-dimensional measurements is a recurring task, including geometric tomography [18], stereology [10], and surface area

© Sunil Arya and David M. Mount;
licensed under Creative Commons License CC-BY 4.0

estimation for digitized 3D objects [24, 22]. Integral formulas of this form are the essential first steps in efficient sampling processes, such as computing ε-nets [3, 14], since they implicitly define the probability distribution upon which to base the sampling.

Modern applications have increasingly explored non-Euclidean spaces, such as the *Hilbert geometry* and its close relative, the *Funk geometry*. (See Section 2 for definitions.) These geometries arise naturally whenever the domain of interest is a convex set. Examples include the analysis of discrete probability distributions in machine learning, where the domain is the probability simplex [12, 25], analysis of networks through hyperbolic geometry [21], analysis of positive definite matrices in deep learning [20, 23], and lattice-based cryptosystems [8]. Hilbert is more widely studied than Funk, but Faifman argues that for many applications, including computing volumes and areas, the Funk geometry is more natural and yields cleaner results [16].

While the metric properties of these spaces are well-understood in differential geometry [16, 26, 27], computational aspects of these geometries are only beginning to emerge. Recent work has employed the Hilbert geometry for polytope approximation [7, 8], clustering [25], and approximate membership queries [2]. Despite this growing algorithmic interest, the integral geometry of these spaces lacks the computational primitives required for efficient implementations. A major barrier is that the standard definition of surface area (the Holmes–Thompson area [19]) relies on symplectic forms or global integrals involving the polar body, making direct numerical evaluation prohibitively complex for high-dimensional or large-scale settings.

In this paper, we bridge this gap by establishing explicit Cauchy and Crofton formulas for the Funk geometry. Our formulas transform the abstract definitions of Finsler area into simple, concrete geometric quantities, which are readily accessible to discrete computation. Beyond the specific interest in Funk geometry, our results reveal it as a unifying bridge connecting diverse geometric settings.

Our main result is a Funk analog of Cauchy's formula. Let K be a convex body in $\mathbb{R}^d$, and let G be a convex set contained in the interior of K. Our objective is to compute the Holmes–Thompson surface area of G in the Funk geometry induced by K, which we denote by $\text{area}_K^F(G)$. For simplicity, let us assume for now that K's boundary is strictly convex, implying that for each direction $u \in S^{d-1}$, there is a unique boundary point $v_K(u)$ whose supporting hyperplane is orthogonal to u (see Figure 1(a)). Define the *central shadow* $S_K(G, u)$ as the slice of the cone subtended by G at $v_K(u)$, orthogonal to the direction u:

$$S_K(G, u) \;=\; -u^* \cap \text{cone}(G - v_K(u)),$$

where $-u^*$ is the hyperplane tangent to S^{d-1} at $-u$ and $\text{cone}(G - v_K(u))$ is the translation of the subtended cone to the origin (see Figure 1(b)). The measure of this section captures the "visual size" of G as seen from the boundary point $v_K(u)$. Just as the Euclidean formula averages orthogonal shadows, our Funk formula averages these central shadows. We show that G's surface area is proportional to the average area of these shadows, taken over all directions u.

▶ **Theorem 1.1.** *Let G and K be two convex bodies in $\mathbb{R}^d$ such that $G \subset \text{int}(K)$. Then*

$$\text{area}_K^F(G) \;=\; \frac{1}{\omega_{d-1}} \int_{S^{d-1}} \lambda_{d-1}(S_K(G, u))\, d\sigma(u),$$

where σ denotes the rotation-invariant surface measure on S^{d-1}.

This offers a "tomographic interpretation" in the sense that the intrinsic Funk surface area can be recovered solely from these central slices, which capture the visibility of G from the boundary of K. It is interesting to note that, while the Holmes–Thompson surface area

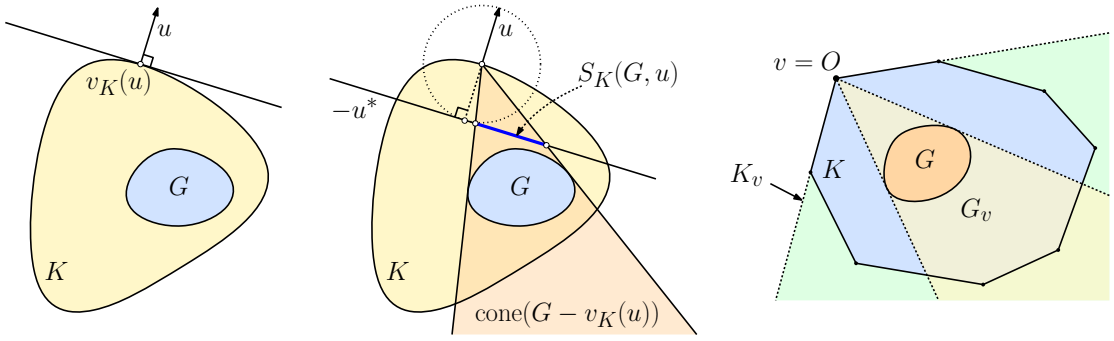


Figure 1 (a) The boundary point $v_K(u)$, (b) the central shadow $S_K(G,u)$ (translated so that $v_K(u)$ coincides with the origin), and (c) vertex decomposition.

of G (presented in Section 2.2) is defined in terms of the areas of the polars of Finsler balls in the Funk geometry induced by K, the formula of Theorem 1.1 involves only simple Euclidean quantities and the standard Lebesgue measure.

Our formulation connects the Funk geometry to several classical settings. When K is expanded to infinity, the central shadows become parallel projections, and our result converges to a Cauchy-type formula for the surface area in Minkowski spaces. In the special case where K is a Euclidean ball, this limit yields the classical Euclidean Cauchy formula. Furthermore, for a finite Euclidean ball K, our formula yields the exact surface area in the Beltrami-Klein model of hyperbolic geometry [13], generalizing planar observations by Alexander *et al.* [5]. Finally, for arbitrary convex bodies, our Funk surface area serves as a constant-factor approximation for the Hilbert surface area, with factors depending only on the dimension d. These results have been omitted due to space restrictions, but they appear in the full version of the paper [9].

Crucially for algorithmic applications, we derive a *discrete surface area formula* when K is a polytope. We show that the Funk surface area of a body G nested within K can be decomposed into a sum of local terms associated with the *vertices of* K. Each vertex v of K is naturally associated with a pointed cone $K_v = \text{cone}(K - v)$, which is the cone subtended by K after translating v to the origin (see Figure 1(c)). The body G is similarly associated with a cone $G_v = \text{cone}(G - v)$. In Section 2.3, we define a simple shadow-based notion of the Funk volume of G_v with respect to K_v, which we denote by $\text{vol}^F_{K_v}(G_v)$. We show that the total surface area of G can be expressed as the sum of these cone-based Funk volumes over the vertices of K.

▶ **Theorem 1.2** (Vertex Decomposition for Polytopes). *Let K be a convex polytope in $\mathbb{R}^d$, and let $G \subset \text{int}(K)$ be a convex body. For each vertex $v \in V(K)$, let $K_v = \text{cone}(K - v)$ and $G_v = \text{cone}(G - v)$. Then*

$$\text{area}^F_K(G) \;=\; \sum_{v \in V(K)} \text{vol}^F_{K_v}(G_v).$$

While the cone-based Funk volume defined in Section 2.3 involves integration, it can be readily estimated through random sampling. Furthermore, this decomposition scales linearly with the vertex count of K (see the discussion following the proof of Theorem 1.1 in Section 4). This is in contrast to prior Crofton measures for Hilbert geometries, which involve quadratic enumerations over face pairs [30].

Finally, we establish a *Funk–Crofton formula* (Theorem 4.4), expressing the Funk surface area as the K-dependent measure of the set of line parameters whose associated lines intersect the body. This result provides the basis for Monte-Carlo estimation of surface area, similar to methods used in Euclidean geometry [1, 22]. That is, one can approximate the Funk surface area simply by sampling from the associated parameter distribution and counting intersections of the corresponding lines with the body, bypassing the need for complex analytical evaluation.

1.1 Related Work

The study of integral geometry in projective Finsler spaces has a rich history in differential geometry [6, 29, 31]. Schneider [29] established the existence of Crofton measures for general projective Finsler spaces using the dual Holmes–Thompson volume. For the specific case of *polytopal Hilbert geometries*, he derived an explicit Crofton measure involving a combinatorial decomposition over pairs of complementary faces of the polytope [30]. Our work differs by focusing on Funk geometry and providing a decomposition based on the *vertices* of the ambient polytope, which as noted above offers significant computational advantages.

In the planar setting, Alexander [4] and later Alexander, Berg, and Foote [5] derived elegant Cauchy-type perimeter formulas for the Hilbert geometry using trigonometric integrals. Note that in this case, Hilbert and Funk surface areas coincide. Like ours, their formulas are explicit and admit a shadow interpretation in the Beltrami-Klein disk model of hyperbolic geometry. However, their techniques are specialized to the plane and do not seem to extend to arbitrary convex domains in higher dimensions. In contrast, our approach unifies these perspectives by showing that the "average of shadows" principle holds for general convex bodies in any dimension, with integrands that are directly interpretable as measures of central projections.

2 Preliminaries

Let us begin by presenting notation and terminology that will be used throughout the paper. We use $\langle \cdot, \cdot \rangle$ for the standard inner (dot) product on $\mathbb{R}^d$, and $\| \cdot \| = \sqrt{\langle \cdot, \cdot \rangle}$ for the Euclidean norm. Let O denote the origin, let B^d be the Euclidean unit ball centered at O, and let $S^{d-1} = \partial B^d$ be the unit sphere. Given a linear subspace $E \subset \mathbb{R}^d$ and a set $K \subset \mathbb{R}^d$, let $K \,|\, E$ denote the orthogonal projection of K onto E.

A *convex body* in $\mathbb{R}^d$ is a compact convex set with nonempty interior. Given a convex set K, we denote its boundary and interior by ∂K and $\mathrm{int}(K)$, respectively. For $\alpha \geq 0$, αK denotes the dilation of K by factor α about the origin, and for $x \in \mathbb{R}^d$, $K + x$ denotes the translate of K by x. Given a convex body K and a direction $u \in S^{d-1}$, the *support function* of K is $h_K(u) = \sup\{\langle u, x \rangle : x \in K\}$, and the corresponding supporting hyperplane is

$$H_K(u) \;=\; \{x \in \mathbb{R}^d : \langle u, x \rangle = h_K(u)\}.$$

For $1 \leq k \leq d$, let λ_k denote the k-dimensional Hausdorff measure on $\mathbb{R}^d$. In particular, λ_d is the Lebesgue measure, and on any C^1 k-dimensional submanifold of $\mathbb{R}^d$, λ_k agrees with the usual k-dimensional surface area. Let σ denote the standard rotation-invariant surface measure on S^{d-1}. Finally, let $\omega_d = \lambda_d(B^d)$.

Given two C^1 hypersurfaces $\Phi, \Psi \subset \mathbb{R}^d$ and a differentiable map $f : \Phi \to \Psi$, the Jacobian of f at $x \in \Phi$ is the local area-stretch factor induced by the differential $Df(x)$ on the tangent space of Φ at x. Equivalently, it is the factor appearing in the change-of-variables formula for surface integrals (see, for example, [15]).

Given two convex sets $A, B \subset \mathbb{R}^d$, their *Minkowski sum* is

$$A + B \;=\; \{a + b \,:\, a \in A,\; b \in B\}.$$

Given a convex body $K \subset \mathbb{R}^d$, its *difference body* is

$$\Delta(K) \;=\; K + (-K),$$

which is centrally symmetric (see Figure 2(a)).

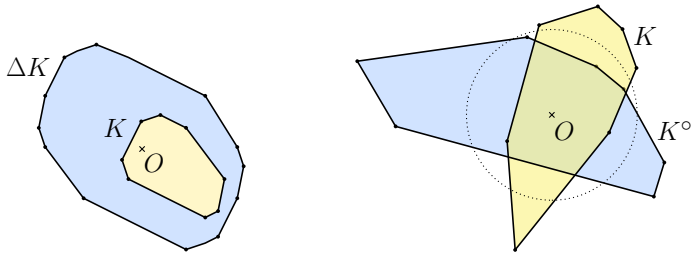


Figure 2 (a) The difference body of a convex body and (b) the polar of a convex body.

For any nonempty set $K \subset \mathbb{R}^d$, its *polar*, denoted K°, is defined by

$$K^\circ \;=\; \{y \in \mathbb{R}^d \,:\, \langle x, y\rangle \le 1 \text{ for all } x \in K\}$$

(see Figure 2(b)). The polar has several standard properties (see, for example, Barvinok [11]). It is always closed, convex, and contains the origin. If $O \in \mathrm{int}(K)$, then K° is bounded. Moreover, if K is a convex body with $O \in \mathrm{int}(K)$, then K° is also a convex body, and $(K^\circ)^\circ = K$. Finally, polarity reverses inclusion: if $K_1 \subseteq K_2$, then $K_2^\circ \subseteq K_1^\circ$.

For a linear subspace $E \subseteq \mathbb{R}^d$ and a nonempty set $G \subseteq E$, define the polar of G in E by

$$G^{\circ_E} \;=\; \{y \in E : \langle x, y\rangle \le 1,\; \text{for all } x \in G\}.$$

The next lemma is the standard duality between sections and projections, stated in the form needed later. A proof may be found in [33, Theorem 2.2.9 and Corollary 2.2.10].

▶ **Lemma 2.1** (Projection-Section Duality). *Let $K \subseteq \mathbb{R}^d$ be a convex body such that $O \in \mathrm{int}(K)$, and let E be a linear subspace of $\mathbb{R}^d$. Then the polar in E of the section $K \cap E$ equals the orthogonal projection of K° onto E. That is,*

$$(K \cap E)^{\circ_E} \;=\; K^\circ \,\big|\, E.$$

2.1 The Funk and Hilbert Geometries

In this section, we introduce the Funk and Hilbert geometries and the related Finsler structures used to define volume and surface area. Let K be a convex body in $\mathbb{R}^d$. For any two distinct points $x, y \in \mathrm{int}(K)$, let y' denote the point where the ray directed from x through y intersects ∂K (see Figure 3(a)). The *Funk distance* with respect to K, denoted $\mathrm{dist}_K^F(\cdot,\cdot)$, is defined to be

$$\mathrm{dist}_K^F(x, y) \;=\; \ln \frac{\|x - y'\|}{\|y - y'\|},$$

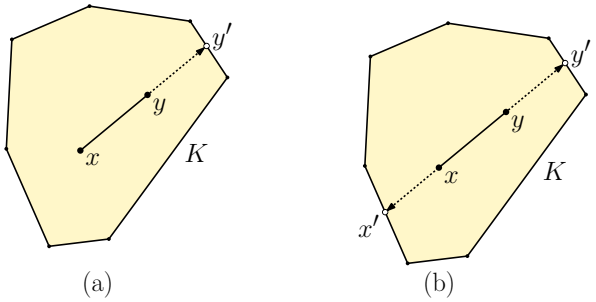


Figure 3 (a) The Funk distance and (b) the Hilbert distance.

where $\mathrm{dist}^F_K(x,y) = 0$ if $x = y$. The Funk distance is nonnegative, asymmetric, satisfies the triangle inequality, and is invariant under invertible affine transformations [27]. Because of its asymmetry, this is often referred to as a weak metric or quasi-metric.

The *Hilbert distance* with respect to K, denoted $\mathrm{dist}^H_K(\cdot,\cdot)$, is the symmetrization of the Funk distance. Letting x' denote the point where the directed ray from y through x intersects ∂K, it is defined to be

$$\mathrm{dist}^H_K(x,y) \;=\; \frac{1}{2}\Big(\mathrm{dist}^F_K(x,y) + \mathrm{dist}^F_K(y,x)\Big) \;=\; \frac{1}{2}\ln\frac{\|y-x'\|}{\|x-x'\|}\frac{\|x-y'\|}{\|y-y'\|}$$

(see Figure 3(b)). It is well known that this is a metric (symmetric, positive except when $x = y$, and satisfying the triangle inequality). Observe that the quantity in the logarithm is the cross ratio $(x, y; y', x')$, and hence the Hilbert distance is invariant under invertible projective transformations.

Before defining volume and surface areas, let us first introduce some related concepts. Given a convex body D in $\mathbb{R}^d$ with $O \in \mathrm{int}(D)$ and $u \in \mathbb{R}^d$, the *Minkowski functional* (or *gauge*) induced by D is defined by

$$\|u\|_D \;=\; \inf\{\lambda > 0 \,:\, u \in \lambda D\}.$$

This functional is positively homogeneous and subadditive. If D is centrally symmetric, then $\|\cdot\|_D$ is a norm in the usual sense; in general, it is the Minkowski functional (or gauge) of D. A *Finsler metric* on a manifold is a continuous function on its tangent bundle, whose restriction to each tangent space is such a gauge. It is well known that both the Funk and Hilbert geometries induce a Finsler structure [34].

To define the Finsler structure for Funk, consider any $x \in \mathrm{int}(K)$. Identify the tangent space at x, denoted T_x, with $\mathbb{R}^d$. For each nonzero $v \in T_x$, let x^+ denote the point where the ray emanating from x in direction v intersects ∂K (see Figure 4(a)), and let t be a positive scalar such that $x + tv = x^+$. The resulting Finsler gauge at x is defined as

$$\mathrm{Finsler}^F_K(x,v) \;=\; \frac{1}{t} \quad\text{and}\quad \mathrm{Finsler}^F_K(x,0) \;=\; 0.$$

This defines a gauge whose unit ball, denoted $B^F_K(x)$, is $K - x$. Thus, the ball is just a translate of K such that x coincides with the origin.

The Finsler structure for Hilbert is a symmetric variant of the Funk Finsler structure. For each $x \in \mathrm{int}(K)$ and each nonzero vector $v \in T_x$, let x^+ be as before, and let x^- be the point where the ray emanating from x in direction $-v$ intersects ∂K (see Figure 4(a)). Let t^+ and t^- be positive scalars such that $x + t^+v = x^+$ and $x - t^-v = x^-$. Then

$$\mathrm{Finsler}^H_K(x,v) \;=\; \frac{1}{2}\left(\frac{1}{t^+} + \frac{1}{t^-}\right) \quad\text{and}\quad \mathrm{Finsler}^H_K(x,0) \;=\; 0.$$

Since this gauge is symmetric, it is a norm. Its unit ball, denoted $B^H_K(x)$, is centrally symmetric (see Figure 4(b)). If K is a polytope, then so is $B^H_K(x)$.

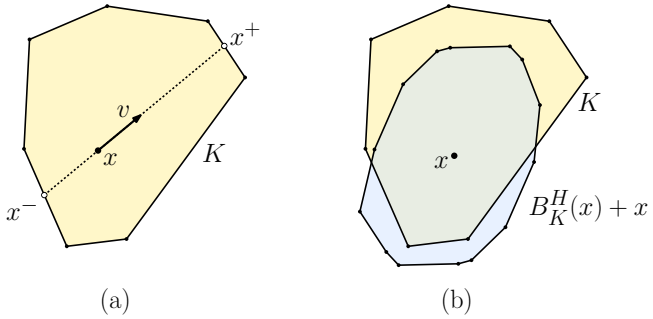


Figure 4 (a) The Finsler structure and (b) the Finsler ball for Hilbert (recentered about x).

The following lemma relates the polar of the Finsler ball in the Hilbert metric to the difference body of $(K - x)^\circ$. The lemma is a straightforward consequence of the Finsler interpretation of the Hilbert metric (see, e.g., [26, 31]).

▶ **Lemma 2.2.** *Given a convex body $K \subset \mathbb{R}^d$ and $x \in \text{int}(K)$, $B_K^H(x)^\circ = \frac{1}{2}\Delta((K - x)^\circ)$.*

2.2 Holmes–Thompson Volume and Surface Area

There are several ways to define volume and surface area in Finsler spaces. The volume of a subset $G \subset \text{int}(K)$ arises by integrating the weighted contributions of volume elements over G, where the weight of each element depends on the local geometry, as expressed through the Finsler ball. Intuitively, as the size of the Finsler ball decreases, the relative importance of the associated Lebesgue volume element increases. Due to the reciprocal nature of polarity, as the volume of the Finsler ball decreases, the volume of its polar increases. This gives rise to the following definitions, due to Holmes and Thompson [19].

Let us first consider the Funk geometry. Given a convex body K and $x \in \text{int}(K)$, the Holmes–Thompson volume element is obtained by scaling the Lebesgue volume element $d\lambda_d(x)$ by the ratio of the Lebesgue volume of the polar of the Funk Finsler ball, $\lambda_d\left(B_K^F(x)^\circ\right)$, and the Lebesgue volume of the unit Euclidean ball, ω_d [17, 19]. Define the *Funk volume element* to be

$$d\,\text{vol}_K^F(x) \;=\; \frac{1}{\omega_d}\lambda_d\left(B_K^F(x)^\circ\right)\, d\lambda_d(x).$$

Recall that $B_K^F(x) = K - x$. The *Funk volume* of any convex body $G \subset \text{int}(K)$ is

$$\text{vol}_K^F(G) \;=\; \int_{x\in G} d\,\text{vol}_K^F(x).$$

The Holmes–Thompson surface area in the Funk geometry is defined analogously. At each smooth boundary point $x \in \partial G$, let T_x denote the tangent space to ∂G at x, viewed as a linear subspace of $\mathbb{R}^d$ and equipped with the induced Hausdorff measure. The *Funk surface-area element* is given by

$$d\,\text{area}_K^F(x) \;=\; \frac{1}{\omega_{d-1}}\lambda_{d-1}\left(\left(B_K^F(x) \cap T_x\right)^{\circ_{T_x}}\right)\, d\lambda_{d-1}(x).$$

By Lemma 2.1, we can express this alternatively in terms of the projected polar as

$$d\,\text{area}_K^F(x) \;=\; \frac{1}{\omega_{d-1}}\lambda_{d-1}\left(B_K^F(x)^\circ \,\middle|\, T_x\right)\, d\lambda_{d-1}(x).$$

Since $B_K^F(x) = K - x$, this is exactly

$$d\,\mathrm{area}_K^F(x) \;=\; \frac{1}{\omega_{d-1}}\lambda_{d-1}\left((K-x)^\circ \;\middle|\; T_x\right)\, d\lambda_{d-1}(x). \tag{2}$$

The *Funk surface area* of $G \subset \mathrm{int}(K)$ is

$$\mathrm{area}_K^F(G) \;=\; \int_{x\in\partial G} d\,\mathrm{area}_K^F(x).$$

The Holmes–Thompson volume and surface area in the Hilbert geometry are defined analogously, replacing the Funk Finsler ball $B_K^F(x)$ by the Hilbert Finsler ball $B_K^H(x)$. We denote the resulting quantities by $\mathrm{vol}_K^H(G)$ and $\mathrm{area}_K^H(G)$.

Faifman showed that the Hilbert- and Funk-based volumes and surface areas are related up to factors that depend on dimension.

▶ **Lemma 2.3** (Faifman [16]). *Let $K \subset \mathbb{R}^d$ be a convex body, let $G \subset \mathrm{int}(K)$ be a convex body, and let $\beta(d) = \binom{2d}{d}/2^d$. Then*

$$\begin{aligned}\mathrm{vol}_K^F(G) \;&\le\; \mathrm{vol}_K^H(G) \;\le\; \beta(d)\cdot \mathrm{vol}_K^F(G) \qquad \textit{and}\\ \mathrm{area}_K^F(G) \;&\le\; \mathrm{area}_K^H(G) \;\le\; \beta(d-1)\cdot \mathrm{area}_K^F(G).\end{aligned}$$

2.3 Cones

The standard Cauchy formula relies on orthogonal projections. We shall see that in the context of the Funk geometry induced by a convex body, the appropriate generalization utilizes central projections towards the boundary of the body. Such projections naturally involve cones. Throughout this paper, a *cone* in $\mathbb{R}^d$ denotes a full-dimensional convex set closed under positive scaling (i.e., if x is in the set, then λx is in the set for all $\lambda \ge 0$). We assume our cones are *pointed*, meaning that they contain no lines. This implies that each cone has a unique *apex* at the origin.

The *conical hull* of a convex set $G \subset \mathbb{R}^d$, denoted $\mathrm{cone}(G)$, is the smallest cone containing G, that is,

$$\mathrm{cone}(G) \;=\; \{\gamma x \,:\, x \in G \text{ and } \gamma \ge 0\}$$

(see Figure 5(a)). Let $K \subset \mathbb{R}^d$ be a closed, full-dimensional convex set. For any boundary point $v \in \partial K$, the *normal cone* at v consists of all outer normal vectors to K at v:

$$\{u \in \mathbb{R}^d : \langle u, x - v\rangle \le 0, \text{ for all } x \in K\}.$$

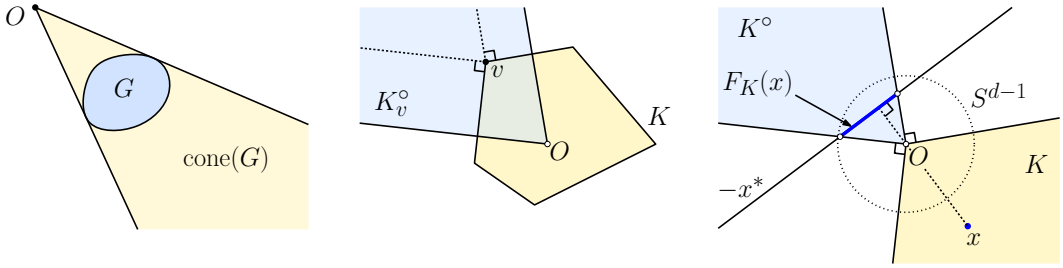
(see Figure 5(b)). In the cases of primary interest to us, such as when v is a vertex of a polytope or the apex of a pointed cone, the tangent cone $K_v = \mathrm{cone}(K - v)$ is pointed. In these settings, the normal cone is full-dimensional and coincides with the polar cone K_v°.

For any nonzero vector $z \in \mathbb{R}^d$, we denote the associated *dual hyperplane* by

$$z^* = \{u \in \mathbb{R}^d \,:\, \langle u, z\rangle = 1\}.$$

For a pointed cone $K \subset \mathbb{R}^d$ and a vector $x \in \mathrm{int}(K)$, we define the *dual cross-section* $F_K(x)$ to be the intersection of the polar cone K° with the dual hyperplane associated with $-x$ (see Figure 5(c)). That is,

$$F_K(x) \;:=\; K^\circ \cap -x^* \;=\; K^\circ \cap (-x)^* \;=\; \{y \in K^\circ : \langle x, y\rangle = -1\}. \tag{3}$$



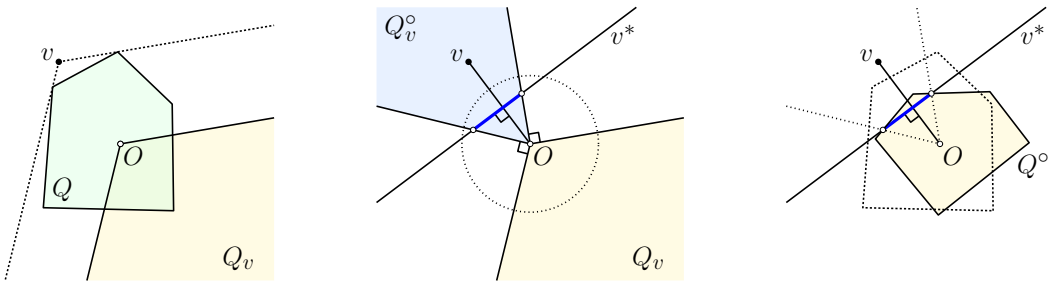
(a) (b) (c)

Figure 5 (a) The cone cone(G), (b) the normal cone K_v°, and (c) $F_K(x)$.

Geometrically, this is the slice of the polar cone by a hyperplane orthogonal to the direction x. Since x lies in the interior of K, and every vector in K° has nonpositive inner product with every vector in K, this hyperplane intersects K° in a bounded set.

Next, we establish a simple technical lemma relating the polar of a body to the polar of its subtended cones. This result allows us to characterize the facets of the polar polytope using local cone geometry.

▶ **Lemma 2.4.** *Let $Q \subset \mathbb{R}^d$ be a convex body containing the origin in its interior. Let $v \in \mathbb{R}^d \setminus \text{int}(Q)$. Let $Q_v = \text{cone}(Q - v)$ be the cone subtended by Q at v (see Figure 6(a)). Then the intersection of the polar cone Q_v° with the dual hyperplane $v^* = \{y \in \mathbb{R}^d : \langle v, y\rangle = 1\}$ coincides with the slice of the polar body Q° by v^*, that is, $Q_v^\circ \cap v^* = Q^\circ \cap v^*$ (see Figure 6(b) and (c)).*



(a) (b) (c)

Figure 6 (a) The cone Q_v, (b) $Q_v^\circ \cap v^*$, and (c) $Q^\circ \cap v^*$.

Proof. By definition, a vector $y \in Q_v^\circ$ if and only if $\langle z, y\rangle \leq 0$ for all $z \in Q_v$. Since Q_v is the conical hull of $Q - v$, this condition is equivalent to:

$$\langle x - v, y\rangle \;\leq\; 0, \quad \forall x \in Q.$$

Restricting our attention to the hyperplane v^*, where $\langle v, y\rangle = 1$, this inequality becomes:

$$\langle x, y\rangle - \langle v, y\rangle \leq 0 \iff \langle x, y\rangle - 1 \leq 0 \iff \langle x, y\rangle \leq 1, \quad \forall x \in Q.$$

The condition $\langle x, y\rangle \leq 1$ for all $x \in Q$ is exactly the definition of the polar body Q°. Thus, restricted to the hyperplane v^*, the condition for membership in the polar cone is identical to the condition for membership in the polar body. ◀

When applied to the vertices of a polytope, this lemma provides a functional characterization of the boundary of the polar body. Recall that for a polytope K with $O \in \operatorname{int}(K)$, the facets of the polar body K° are in one-to-one correspondence with the vertex set $V(K)$ [35]. Specifically, for each vertex $v \in V(K)$, the dual facet is the intersection of K° with the supporting hyperplane v^*. Lemma 2.4 implies that this facet is exactly $K_v^\circ \cap v^*$, which matches the definition of the dual cross-section $F_{K_v}(-v)$.

▶ **Corollary 2.5.** *Let $K \subset \mathbb{R}^d$ be a convex polytope with $O \in \operatorname{int}(K)$. For each vertex $v \in V(K)$, the facet of K° dual to v, denoted F_v, is given by*

$$F_v \;=\; K_v^\circ \cap v^* \;=\; F_{K_v}(-v).$$

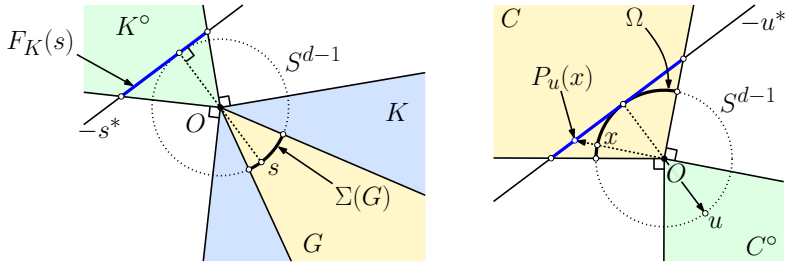
Furthermore, $\partial K^\circ = \bigcup_{v \in V(K)} F_v$.

Extending the definition of Funk volume from bounded convex bodies to unbounded cones requires care. If G and K are pointed cones in $\mathbb{R}^d$ with $(G \setminus \{O\}) \subset \operatorname{int}(K)$, the standard Funk distance is degenerate, and the classical volume is infinite. However, the projective nature of Funk geometry allows us to define a meaningful volume for G relative to K by considering cross-sections. Faifman [16] demonstrated that the Funk geometry is essentially projective. A key consequence is that the Holmes–Thompson volume of the section $G \cap H$, measured with respect to the ambient section $K \cap H$, is invariant to the choice of the hyperplane H, provided $K \cap H$ is bounded. We call such a hyperplane *admissible*.

This invariance implies that the volume is intrinsic to the nested cone structure itself. We define the *cone Funk volume* of G relative to K by the integral

$$\operatorname{vol}_K^F(G) \;:=\; \frac{1}{\omega_{d-1}} \int_{\Sigma(G)} \lambda_{d-1}(F_K(s))\, d\sigma(s), \qquad \text{where } \Sigma(G) := S^{d-1} \cap G \tag{4}$$

(see Figure 7(a)). In the full version of the paper [9], we show that this integral coincides with $\operatorname{vol}_{K\cap H}^F(G \cap H)$ for any admissible hyperplane H, thus justifying this terminology.



(a) (b)

Figure 7 (a) Defining $\operatorname{vol}_K^F(G)$ and (b) the gnomonic projection.

When dealing with spherical cross-sections of cones, it is useful to relate their spherical measures to the Euclidean measures of corresponding hyperplane sections. Let $C \subset \mathbb{R}^d$ be a pointed cone, let $\Omega = C \cap S^{d-1}$, and let $u \in S^{d-1}$ satisfy $\langle x, u \rangle < 0$ for all $x \in \Omega$. The associated gnomonic projection maps Ω onto the hyperplane section $-u^* \cap C$ (see Figure 7(b)). A straightforward Jacobian computation yields the following lemma (see, e.g., [16, Lemma 3.10]).

▶ **Lemma 2.6** (Cone gnomonic projection). *Let $C \subset \mathbb{R}^d$ be a pointed cone, let $\Omega = C \cap S^{d-1}$, and let $u \in S^{d-1}$ satisfy $\langle x, u\rangle < 0$, for all $x \in \Omega$ (equivalently, $u \in \mathrm{int}(C^\circ)$). Define*

$$P_u(x) \;:=\; -\frac{x}{\langle x, u\rangle} \qquad (x \in \Omega).$$

Then $P_u(\Omega) = -u^* \cap C$, and

$$\lambda_{d-1}(P_u(\Omega)) \;=\; \int_\Omega |\,\langle x, u\rangle\,|^{-d}\, d\sigma(x).$$

3 Vertex Decomposition for Convex Polytopes

In this section, we establish a discrete surface area formula for the case where K is a convex polytope. Our result provides a vertex-based decomposition of the Funk surface area similar in spirit to that of Alexander *et al.* [5] in the planar Hilbert geometry, but derived by a distinct approach that applies in arbitrary dimensions.

Our proof proceeds by decomposing the Funk surface area of G into local contributions associated with the vertices of the ambient polytope K. For each vertex $v \in V(K)$, let $K_v = \mathrm{cone}(K - v)$ and $G_v = \mathrm{cone}(G - v)$ denote the local cones subtended at v. We will show that the total Funk surface area of G is the sum of the Funk volumes of the cones G_v relative to K_v. We begin with a lemma that expresses the Funk volume of $\mathrm{cone}(G)$ relative to a pointed cone K as a boundary integral involving the projected areas of the dual cross-sections $F_K(x)$. In the statement below, T_x denotes the tangent space of G at x.

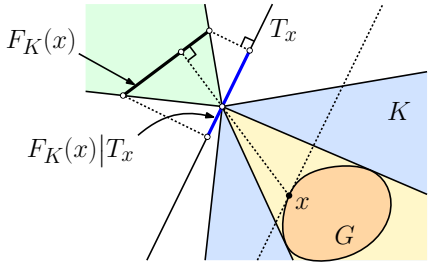


Figure 8 Lemma 3.1.

▶ **Lemma 3.1.** *Let $K \subset \mathbb{R}^d$ be a pointed cone and let $G \subset \mathrm{int}(K)$ be a convex body. Then*

$$2 \cdot \mathrm{vol}_K^F(\mathrm{cone}(G)) \;=\; \frac{1}{\omega_{d-1}} \int_{\partial G} \lambda_{d-1}\big(F_K(x) \,\big|\, T_x\big)\, d\lambda_{d-1}(x)$$

(see Figure 8).

The next step in our derivation utilizes the dual facet characterization (Corollary 2.5) to decompose the projection of the polar body into a sum over its facets.

▶ **Lemma 3.2.** *Let $K \subset \mathbb{R}^d$ be a convex polytope. For any $x \in \mathrm{int}(K)$ and any $(d-1)$-dimensional linear subspace T,*

$$2\lambda_{d-1}\big((K - x)^\circ \,\big|\, T\big) \;=\; \sum_{v \in V(K)} \lambda_{d-1}\big(F_{K_v}(x - v) \,\big|\, T\big)$$

(see Figures 9(a) and (b)).

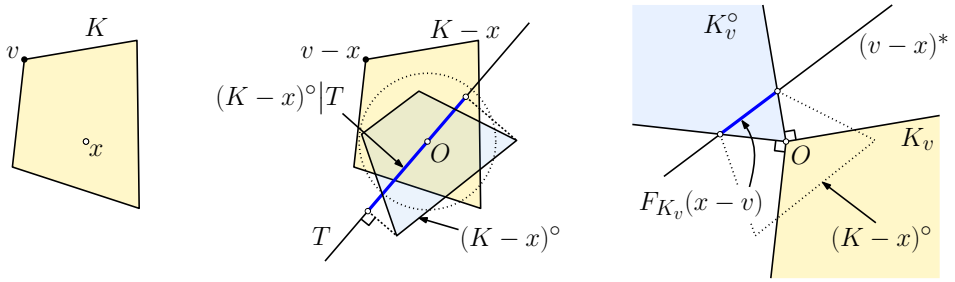


Figure 9 Lemma 3.2 and its proof.

The proofs of Lemmas 3.1 and 3.2 appear in the full version of the paper [9].
We are now ready to prove the vertex decomposition theorem from Section 1.

Proof. (Of Theorem 1.2) For each $v \in V(K)$, applying Lemma 3.1 to the pair $(K_v, G - v)$ yields

$$2 \cdot \mathrm{vol}^F_{K_v}(G_v) \;=\; \frac{1}{\omega_{d-1}} \int_{\partial(G-v)} \lambda_{d-1}\left(F_{K_v}(x') \,\middle|\, T_{x'}\right)\, d\lambda_{d-1}(x').$$

Perform the change of variables $x = x' + v$. Then $\partial(G - v)$ maps to ∂G, and the tangent spaces are identified as $T_{x'} = T_x$, yielding

$$2 \cdot \mathrm{vol}^F_{K_v}(G_v) \;=\; \frac{1}{\omega_{d-1}} \int_{\partial G} \lambda_{d-1}\left(F_{K_v}(x - v) \,\middle|\, T_x\right)\, d\lambda_{d-1}(x).$$

Summing over $v \in V(K)$ and using linearity of integration, we have

$$2 \sum_{v \in V(K)} \mathrm{vol}^F_{K_v}(G_v) \;=\; \frac{1}{\omega_{d-1}} \int_{\partial G} \left[\sum_{v \in V(K)} \lambda_{d-1}\left(F_{K_v}(x - v) \,\middle|\, T_x\right) \right] d\lambda_{d-1}(x).$$

By Lemma 3.2, the term in brackets is equal to $2\,\lambda_{d-1}((K - x)^\circ \,|\, T_x)$. Substituting this and dividing by 2 yields

$$\sum_{v \in V(K)} \mathrm{vol}^F_{K_v}(G_v) \;=\; \frac{1}{\omega_{d-1}} \int_{\partial G} \lambda_{d-1}\left((K - x)^\circ \,\middle|\, T_x\right)\, d\lambda_{d-1}(x).$$

By Eq. (2), the right-hand side is exactly $\mathrm{area}^F_K(G)$. ◀

4 Cauchy-Type Formulas for General Convex Bodies

We now derive the general Cauchy formula from the polytope case. The key additional ingredient is a representation of the Funk volume of a cone as an average of the areas of its central shadows. Combined with Theorem 1.2, this yields the formula for polytopal K, and the general case then follows by approximation.

▶ **Lemma 4.1** (Funk Volume of a Cone). *Let K and G be pointed cones in $\mathbb{R}^d$ with $(G \setminus \{O\}) \subset \mathrm{int}(K)$. Let $U = K^\circ \cap S^{d-1}$ (the spherical image of the polar cone), and for $u \in U$, let $S(u) = -u^* \cap G$ (see Figure 10(a)). Then*

$$\mathrm{vol}^F_K(G) \;=\; \frac{1}{\omega_{d-1}} \int_U \lambda_{d-1}(S(u))\, d\sigma(u). \tag{5}$$

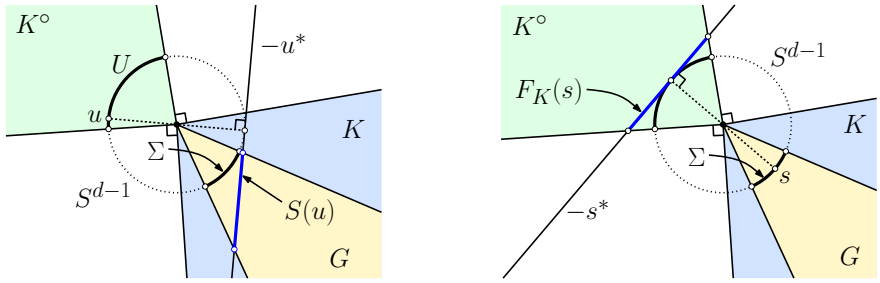


Figure 10 Lemma 4.1 and its proof.

Proof. Let $\Sigma = S^{d-1} \cap G$ and define the double integral

$$I \;=\; \frac{1}{\omega_{d-1}} \int_U \int_\Sigma |\langle s, u\rangle|^{-d}\, d\sigma(s)\, d\sigma(u).$$

Since the integrand is nonnegative, by Tonelli's theorem [28, Theorem 8.8(a)], we may integrate in either order. We will show that one order leads to the right-hand side of Eq. (5), and the other leads to the left-hand side.

First fix $u \in U$ and consider the inner integral in s. Since $u \in K^\circ$ and $(G \setminus \{O\}) \subset \mathrm{int}(K)$, we have $\langle s, u\rangle < 0$ for all $s \in \Sigma$. Applying Lemma 2.6 to the cone G with center u, we obtain

$$\int_\Sigma |\langle s, u\rangle|^{-d}\, d\sigma(s) \;=\; \lambda_{d-1}(-u^* \cap G) \;=\; \lambda_{d-1}(S(u))$$

(see Figure 10(a)). Substituting into I yields the right-hand side of Eq. (5).

Next fix $s \in \Sigma$ and consider the inner integral in u. Since $s \in \Sigma \subset \mathrm{int}(K)$ and $U = K^\circ \cap S^{d-1}$, we have $\langle u, s\rangle < 0$ for all $u \in U$. Applying Lemma 2.6 to the cone K° with center s, we obtain

$$\int_U |\langle u, s\rangle|^{-d}\, d\sigma(u) \;=\; \lambda_{d-1}(-s^* \cap K^\circ) \;=\; \lambda_{d-1}(F_K(s))$$

(see Figure 10(b)). Substituting into I, we obtain

$$I \;=\; \frac{1}{\omega_{d-1}} \int_\Sigma \lambda_{d-1}(F_K(s))\, d\sigma(s).$$

By the definition of Funk volume (Eq. (4)), this expression is exactly $\mathrm{vol}_K^F(G)$, which matches the left-hand side of Eq. (5). ◀

With the cone formula established, we now derive the Cauchy formula for arbitrary convex bodies. Let $K \subset \mathbb{R}^d$ be a convex body and let $G \subset \mathrm{int}(K)$ be a convex body. For almost every direction $u \in S^{d-1}$, the support set $K \cap H_K(u)$ is a singleton; denote its unique point by $v_K(u)$ [32, Theorem 2.2.11]. On the exceptional null set, choose $v_K(u)$ arbitrarily in $K \cap H_K(u)$. For each $u \in S^{d-1}$, define the central shadow by

$$S_K(G, u) \;:=\; -u^* \cap \mathrm{cone}(G - v_K(u))$$

(recall Figure 1). We are now ready to prove Theorem 1.1.

Proof of Theorem 1.1. Assume first that K is a convex polytope. For each vertex $v \in V(K)$, let

$$U_v \;:=\; K_v^\circ \cap S^{d-1}$$

denote the spherical image of the normal cone at v. The collection $\{U_v\}_{v\in V(K)}$ forms a partition of S^{d-1} up to a set of σ-measure zero. For almost every $u \in U_v$, the support point of K in direction u is unique and equals v. Recalling that $G_v = \mathrm{cone}(G - v)$, it follows that for almost every $u \in U_v$,

$$S_{K_v}(G_v, u) \;=\; S_K(G, u).$$

Therefore, by Theorem 1.2 and Lemma 4.1, we obtain

$$\begin{aligned}
\mathrm{area}_K^F(G) \;&=\; \sum_{v\in V(K)} \mathrm{vol}_{K_v}^F(G_v) \\
&=\; \sum_{v\in V(K)} \frac{1}{\omega_{d-1}} \int_{U_v} \lambda_{d-1}(S_{K_v}(G_v, u))\, d\sigma(u) \\
&=\; \frac{1}{\omega_{d-1}} \sum_{v\in V(K)} \int_{U_v} \lambda_{d-1}(S_K(G, u))\, d\sigma(u) \\
&=\; \frac{1}{\omega_{d-1}} \int_{S^{d-1}} \lambda_{d-1}(S_K(G, u))\, d\sigma(u).
\end{aligned}$$

This establishes the result when K is a polytope. The general case follows by approximating K in the Hausdorff metric by convex polytopes K_m, applying the polytopal identity to K_m, and passing to the limit using the continuity of polarity and the Dominated Convergence Theorem; details appear in the full version [9]. ◀

From a computational perspective, when K is a polytope, the preceding proof yields a natural Monte-Carlo estimator. The identity

$$\mathrm{area}_K^F(G) \;=\; \sum_{v\in V(K)} \frac{1}{\omega_{d-1}} \int_{U_v} \lambda_{d-1}(S_{K_v}(G_v, u))\, d\sigma(u)$$

decomposes the surface area into local contributions indexed by the vertices of K. The sampling distribution depends only on the spherical normal-cone decomposition $\{U_v\}_{v\in V(K)}$, while the sampled quantity is the $(d-1)$-dimensional volume of the corresponding central shadow. Thus one may sample a vertex v with probability proportional to $\sigma(U_v)$ and then sample a direction u uniformly from U_v, for example by triangulating the normal cone into simplicial cones. This gives an unbiased estimator whose evaluation uses only geometry local to the chosen vertex. In particular, it avoids direct use of the Holmes–Thompson definition through Eq. (2), which would require integrating

$$\lambda_{d-1}\big((K - x)^\circ \,\big|\, T_x\big)$$

over all $x \in \partial G$.

Using Lemma 2.6, we can rewrite the shadow integral in Theorem 1.1 in terms of the spherical cross-sections of the subtended cones. For any $v \in \partial K$, let

$$\Sigma(G, v) \;:=\; S^{d-1} \cap \mathrm{cone}(G - v)$$

denote the spherical cross-section of the cone subtended by G at v. The following corollary gives a double-integral representation of the Funk surface area.

▶ **Corollary 4.2** (Double-Integral Formula for Funk Area). *Let G and K be convex bodies in $\mathbb{R}^d$ with $G \subset \mathrm{int}(K)$. Then*

$$\mathrm{area}_K^F(G) \;=\; \frac{1}{\omega_{d-1}} \int_{S^{d-1}} \int_{\Sigma(G, v_K(u))} |\langle s, u\rangle|^{-d}\, d\sigma(s)\, d\sigma(u).$$

Proof. By Theorem 1.1,

$$\mathrm{area}_K^F(G) \;=\; \frac{1}{\omega_{d-1}} \int_{S^{d-1}} \lambda_{d-1}(S_K(G,u))\, d\sigma(u).$$

For each $u \in S^{d-1}$,

$$S_K(G,u) \;=\; -u^* \cap \mathrm{cone}(G - v_K(u)).$$

Since $v_K(u)$ is a support point of K in direction u and $G \subset \mathrm{int}(K)$, we have $\langle s, u\rangle < 0$ for all $s \in \Sigma(G, v_K(u))$. Applying Lemma 2.6 with $C = \mathrm{cone}(G - v_K(u))$ and center u, we obtain

$$\lambda_{d-1}(S_K(G,u)) \;=\; \int_{\Sigma(G, v_K(u))} |\langle s, u\rangle|^{-d}\, d\sigma(s).$$

Substituting this into the preceding formula completes the proof. ◀

We now recast Corollary 4.2 as a Crofton formula on oriented lines. For each $u \in S^{d-1}$, choose $v_K(u) \in K \cap H_K(u)$ as above, and let

$$\mathcal{P} \;:=\; \{(u,s) \in S^{d-1} \times S^{d-1} ST\, \langle s, u\rangle < 0\}.$$

For $(u,s) \in \mathcal{P}$, let $\ell_K^+(u,s)$ denote the oriented line through $v_K(u)$ in direction s. For fixed $u \in S^{d-1}$, a direction $s \in S^{d-1}$ with $\langle s, u\rangle < 0$ belongs to $\Sigma(G, v_K(u))$ if and only if the ray from $v_K(u)$ in direction s meets G. Thus Corollary 4.2 can be rewritten as follows.

▶ **Lemma 4.3** (Funk–Crofton formula for oriented lines). *Let G and K be convex bodies in $\mathbb{R}^d$ with $G \subset \mathrm{int}(K)$. Then*

$$\mathrm{area}_K^F(G) \;=\; \frac{1}{\omega_{d-1}} \int_{\mathcal{P}} \mathbf{1}_{\{\ell_K^+(u,s) \cap G \neq \emptyset\}}\, |\, \langle s, u\rangle\, |^{-d}\, d\sigma(s)\, d\sigma(u).$$

Replacing each oriented line by its underlying unoriented line, we obtain the following K-dependent parameter-space Crofton formula.

▶ **Theorem 4.4** (Funk–Crofton formula for unoriented lines). *Let G and K be convex bodies in $\mathbb{R}^d$ with $G \subset \mathrm{int}(K)$. Then*

$$\mathrm{area}_K^F(G) \;=\; \frac{1}{\omega_{d-1}} \int_{\mathcal{P}} \mathbf{1}_{\{L_K(u,s) \cap G \neq \emptyset\}}\, |\, \langle s, u\rangle\, |^{-d}\, d\sigma(s)\, d\sigma(u),$$

where $L_K(u,s)$ denotes the unoriented line underlying $\ell_K^+(u,s)$.

Proof. Since intersection with G is independent of orientation, the indicator in Lemma 4.3 is unchanged when $\ell_K^+(u,s)$ is replaced by its underlying unoriented line $L_K(u,s)$. ◀

Concluding Remarks and Open Problems

We established an explicit Cauchy-type formula for the Holmes–Thompson surface area in the Funk geometry induced by a convex body $K \subset \mathbb{R}^d$. This formula expresses the surface area of a convex body G as the average, over directions $u \in S^{d-1}$, of the $(d-1)$-dimensional measures of the corresponding central shadows of G (Theorem 1.1). For polytopal K, this identity admits a discrete vertex-based decomposition (Theorem 1.2), and the same framework yields a Crofton-type representation in terms of an explicit measure on the space of unoriented lines intersecting G (Theorem 4.4).

Taken together, these results provide a concrete shadow-averaging principle for surface area in a projective Finsler setting. From a computational perspective, our formulas involve Euclidean $(d-1)$-dimensional volumes of explicitly defined slices or projections, thereby avoiding direct evaluation of the Holmes–Thompson definitions through polars of pointwise Finsler balls. In the polytopal case, the resulting decomposition is especially simple, replacing more combinatorial Crofton descriptions based on pairs of faces by a sum over vertices and normal cones.

Our work raises several natural open problems. A first direction is to develop provably efficient randomized algorithms for estimating Funk surface area from the Cauchy and Crofton formulas, ideally with explicit variance bounds and high-probability guarantees. For polytopal K, this includes efficient sampling from the spherical normal-cone decomposition and efficient evaluation or approximation of the associated central shadows. It would also be interesting to determine to what extent these formulas can serve as practical primitives for geometric computation in Funk- and Hilbert-type domains, in a manner analogous to the role of Cauchy and Crofton formulas in Euclidean stereology, tomography, and randomized surface area estimation.

A second direction is to investigate whether this shadow-based approach extends beyond surface area to higher-order intrinsic quantities, such as the Holmes–Thompson analogs of quermassintegrals or curvature measures. A third is to better understand the relationship with Hilbert geometry, especially by identifying conditions on K under which the Funk formula yields the exact Hilbert surface area beyond the known cases of $d = 2$ and ellipsoids. More generally, it would be interesting to determine how geometric properties of K govern the approximation gap between Hilbert and Funk surface areas.

References

1 C. Aaron, A. Cholaquidis, and R. Fraiman. Estimation of surface area. *Electronic Journal of Statistics*, 16(2):3751–3788, 2022. doi:10.1214/22-EJS2031.

2 A. Abdelkader and D. M. Mount. Convex approximation and the Hilbert geometry. In *Proc. SIAM Symp. Simplicity in Algorithms (SOSA24)*, pages 286–298, 2024. doi:10.1137/1.9781611977936.26.

3 P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. Assoc. Comput. Mach.*, 51:606–635, 2004. doi:10.1145/1008731.1008736.

4 R. Alexander. Planes for which the lines are the shortest paths between points. *Ill. J. Math.*, 22:177–190, 1978. doi:10.1215/ijm/1256048729.

5 R. Alexander, I. D. Berg, and R. L. Foote. Integral-geometric formulas for perimeter in S^2, H^2, and Hilbert planes. *Rocky Mountain J. Mathematics*, 35(6):1825–1859, 2005. URL: http://www.jstor.org/stable/44238783.

6 J. C. Álvarez Paiva and E. Fernandes. Crofton formulas in projective Finsler spaces. *Electron. Res. Announc. Amer. Math. Soc.*, 4:91–100, 1998. URL: http://eudml.org/doc/224916.

7 R. Arya, S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal bound on the combinatorial complexity of approximating polytopes. *ACM Trans. Algorithms*, 18:1–29, 2022. doi:10.1145/3559106.

8 S. Arya, G. D. da Fonseca, and D. M. Mount. Economical convex coverings and applications. *SIAM J. Comput.*, 53(4):1002–1038, 2024. doi:10.1137/23M1568351.

9 S. Arya and D. M. Mount. Cauchy's surface area formula in the funk geometry, 2026. doi:10.48550/arXiv.2601.16468.

10 A. Baddeley and E. B. V. Jensen. *Stereology for Statisticians*. CRC Press, 2005. doi:10.1201/9780203496817.

11 A. Barvinok. *A Course in Convexity*, volume 54 of *Graduate Studies in Mathematics*. American Mathematical Society, 2002. URL: https://bookstore.ams.org/gsm-54/.

12 J.-D. Boissonnat, F. Nielsen, and R. Nock. Bregman Voronoi diagrams. *Discrete Comput. Geom.*, 44:281–307, 2010. doi:10.1007/s00454-010-9256-1.

13 J. W. Cannon, W. J. Floyd, R. Kenyon, and W. R. Parry. Hyperbolic geometry. In S. Levy, editor, *Flavors of Geometry*, volume 31 of *MSRI Publications*, pages 59–115. Cambridge University Press, 1997.

14 K. L. Clarkson. Building triangulations using ε-nets. In *Proc. 38th Annu. ACM Sympos. Theory Comput.*, pages 326–335, 2006. doi:10.1145/1132516.1132564.

15 M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

16 D. Faifman. A Funk perspective on billiards, projective geometry and Mahler volume. *J. Differential Geom.*, 127:161–212, 2024. doi:10.4310/jdg/1717356157.

17 D. Faifman, C. Vernicos, and C. Walsh. Volume growth of Funk geometry and the flags of polytopes. *Geometry & Topology*, 29(7):3773–3811, 2025. doi:10.2140/gt.2025.29.3773.

18 R. J. Gardner. *Geometric Tomography*. Cambridge University Press, 2nd edition, 2006. doi:10.1017/CBO9781107341029.

19 R. D. Holmes and A. C. Thompson. n-dimensional area and content in Minkowski spaces. *Pacific J. Math.*, 85:77–110, 1979. doi:10.2140/pjm.1979.85.77.

20 M. Ito, Y. Seo, T. Yamazaki, and M. Yanagida. Geometric properties of positive definite matrices cone with respect to the Thompson metric. *Linear Algebra Appl.*, 435:2054–2064, 2011. doi:10.1016/j.laa.2011.03.063.

21 D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá. Hyperbolic geometry of complex networks. *Phys. Rev. E*, 82, 2010. doi:10.1103/PhysRevE.82.036106.

22 X. Li, W. Wang, R. R. Martin, and A. Bowyer. Using low-discrepancy sequences and the crofton formula to compute surface areas of geometric models. *Computer-Aided Design*, 35(9):771–782, 2003. doi:10.1016/S0010-4485(02)00100-8.

23 W. Lin, V. Duruisseaux, M. Leok, F. Nielsen, M. E. Khan, and M. Schmidt. Simplifying momentum-based positive-definite submanifold optimization with applications to deep learning. In *Proc. 40th Internat. Conf. Mach. Learn.*, 2023. arXiv:2302.09738.

24 Y.-S. Liu, J. Yi, H. Zhang, G.-Q. Zheng, and J.-C. Paul. Surface area estimation of digitized 3d objects using quasi-monte carlo methods. *Pattern Recognition*, 43:3900–3909, 2010. doi:10.1016/j.patcog.2010.06.002.

25 F. Nielsen and K. Sun. Clustering in Hilbert's projective geometry: The case studies of the probability simplex and the elliptope of correlation matrices. In *Geometric Structures of Information*, Signals and communication technology, pages 297–331. Springer International Publishing, 2019. doi:10.1007/978-3-030-02520-5_11.

26 A. Papadopoulos and M. Troyanov. Harmonic symmetrization of convex sets and of Finsler structures, with applications to Hilbert geometry. *Expo. Math.*, 27:109–124, 2009. doi:10.1016/j.exmath.2008.10.001.

27 A. Papadopoulos and M. Troyanov. *Handbook of Hilbert Geometry*. EMS Press, 2014. doi:10.4171/147.

28 W. Rudin. *Real and Complex Analysis*. McGraw Hill, 3rd edition, 1987.

29 R. Schneider. Crofton formulas in hypermetric projective Finsler spaces. *Archiv der Mathematik*, 77:85–97, 2001. doi:10.1007/PL00000469.

30 R. Schneider. Crofton measures in polytopal Hilbert geometries. *Beitr. Algebra Geom.*, 47:479–488, 2006. URL: https://www.emis.de/journals/BAG/vol.47/no.2/12.html.

31 R. Schneider. Crofton measures in projective Finsler spaces. In E. L. Grinberg, S. Li, G. Zhang, and J. Zhou, editors, *Integral Geometry and Convexity*, pages 67–98. World Scientific, 2006. doi:10.1142/9789812774644_0006.

32 R. Schneider. *Convex Bodies: The Brunn-Minkowski Theory*, volume 151 of *Encyclopedia of Mathematics and its Applications*. Cambridge Univ. Press, 2nd edition, 2014. doi:10.1017/CBO9781139003858.

33 A. C. Thompson. *Minkowski Geometry*, volume 63 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 1996. doi:10.1017/CBO9781107325845.

34 M. Troyanov. Funk and Hilbert geometries from the Finslerian viewpoint. In *Handbook of Hilbert geometry*, pages 69–110. European Mathematical Society Publishing House, 2014. doi:10.4171/147-1/3.

35 Günter M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995. doi:10.1007/978-1-4613-8431-1.

Intersection Patterns of Set Systems on Manifolds with Slowly Growing Homological Shatter Functions

Sergey Avvakumov
School of Mathematical Sciences, Tel Aviv University, Israel

Marguerite Bin
Université de Lorraine, CNRS, INRIA, LORIA, F-54000 Nancy, France

Xavier Goaoc
Université de Lorraine, CNRS, INRIA, LORIA, F-54000 Nancy, France

Abstract

A theorem of Matoušek asserts that for any $k \geq 2$, any set system whose shatter function is $o(n^k)$ enjoys a fractional Helly theorem of order k: in the k-wise intersection hypergraph, positive density implies a linear-size clique. Kalai and Meshulam conjectured a generalization of that phenomenon to *homological shatter functions*. It was verified for set systems with bounded homological shatter functions and whose ground set has a *forbidden homological minor* (which includes $\mathbb{R}^d$ by a homological analogue of the van Kampen-Flores theorem). We present two contributions to this line of research:

- We study homological minors in certain manifolds (possibly with boundary), for which we prove analogues of the van Kampen-Flores theorem and of the Hanani-Tutte theorem.
- We introduce graded analogues of the Radon and Helly numbers of set systems and relate their growth rate to the original parameters. This allows to extend the verification of the Kalai-Meshulam conjecture to sufficiently slowly growing homological shatter functions.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Fractional Helly theorem, homological minor, combinatorial convexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.9

Related Version *Full Version*: `https://arxiv.org/abs/2601.02920` [2]

1 Introduction

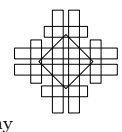
A classical line of research in discrete geometry investigates generalizations of properties of convex sets beyond convexity, with a particular attention to topological conditions. At least three distinct lines of inquiry emerged:

(A) Generalizing families of convex sets into *acyclic/good covers*, meaning set systems in topological spaces such that every subfamily has empty or homologically/homotopically trivial intersection; an early example is Helly's topological theorem [15], see also the survey of Tancer [34] for an overview.

(B) Reformulating properties of convex sets of $\mathbb{R}^d$ as properties of linear maps into $\mathbb{R}^d$ and investigating their generalizations to *continuous* maps into $\mathbb{R}^d$, typically via theorems of Borsuk-Ulam type; an early example is the topological Radon theorem of Bajmóczy and Bárány [3] and more examples can be found *e.g.* in the survey of Bárány and Soberón [5].

(C) Analyzing set systems whose nerve enjoy properties of nerves of convex sets, like *d-collapsibility* or *d-Lerayness*; an early example is the sharpening of the Fractional Helly theorem by Kalai [18] and the work of Alon et al. [1] establishes several landmark results.


© Sergey Avvakumov, Marguerite Bin, and Xavier Goaoc;
licensed under Creative Commons License CC-BY 4.0




Two decades ago, Kalai and Meshulam [19] proposed conjectures relating approaches (A) and (C), towards what they called a theory of *homological VC dimension*. First, in order to generalize the notion of *good cover*, let us measure the complexity of the intersection patterns of a set system $\mathcal{F}$ in a topological space by its *hth homological shatter function*

$$\phi_{\mathcal{F}}^{(h)}: \left\{ \begin{array}{ccc} \mathbb{N} & \to & \mathbb{N} \cup \{\infty\} \\ k & \mapsto & \sup \left\{ \tilde{\beta}_i \left(\bigcap_{F \in \mathcal{G}} F; \mathbb{Z}_2 \right) \;\middle|\; \mathcal{G} \subset \mathcal{F}, |\mathcal{G}| \leq k, 0 \leq i \leq h \right\}. \end{array} \right. \tag{1}$$

Here h is some fixed parameter, and $\tilde{\beta}_i(\cdot; \mathbb{Z}_2)$ is the ith reduced Betti number with coefficients in $\mathbb{Z}_2$. (The set systems $\mathcal{F}$ with $\phi_{\mathcal{F}}^{(\infty)} \equiv 0$ are the acyclic covers and include good covers and convex sets.) The conjectures are about nerves of set systems whose intersection patterns have polynomially-growing topological complexity. In particular, their combination [19, Conjectures 6 and 7] (see also [13, Conjecture 1.9]) implies that polynomially-growing homological shatter functions give rise to a "*positive density implies big clique*" phenomenon:

▶ **Conjecture 1** (Kalai and Meshulam)**.** *For any $d \in \mathbb{N}$ and any function $\Psi : \mathbb{N} \to \mathbb{N}$ such that $\Psi(n) = O(n^d)$, there exists $\beta : (0,1) \to (0,1)$ such that the following holds. For any $\alpha > 0$ and any set system $\mathcal{F}$ in $\mathbb{R}^d$ with $\phi_{\mathcal{F}}^{(d)} \leq \Psi$, if a proportion α of the $(d+1)$-element subsets of $\mathcal{F}$ have nonempty intersection, then some $\beta(\alpha)|\mathcal{F}|$ members of $\mathcal{F}$ have a point in common.*

Such conditions that a positive density of d-faces in the nerve implies the existence of a linear-size face is called a *fractional Helly theorem* (see Section 1.1.1).

Conjecture 1 is a topological analogue of a theorem of Matoušek [25] which asserts that every set system with polynomial (combinatorial) shatter function enjoys a fractional Helly theorem.

Conjecture 1 was confirmed for functions Ψ that are bounded ([13, Corollary 1.3], building on [16, 17, 28]). The main contribution of the present paper is to extend that confirmation to some diverging homological shatter functions and to set systems on certain manifolds.

1.1 Context and motivation

Before we state our results precisely (in Section 1.2) let us provide some context and motivation, as well as introduce some necessary terminology.

1.1.1 Combinatorial background: convexity parameters of set systems

The classical theorems of Helly, Radon and Carathéodory have initiated a rich theory of the combinatorial properties of convexity, whose landmarks include the centerpoint theorem, Tverberg's theorem, the colorful Helly and Carathéodory theorems, the fractional Helly theorem, the selection lemma, the weak ε-net theorem, the (p,q)-theorem, etc. We refer the interested reader to the monograph of Bárány [4] and the textbook of Matoušek [22]. These classical convexity theorems have algorithmic consequences for instance in optimization and geometric data analysis [8, § 6 – 7] or in property testing [7], and one motivation for their extension beyond the convex setting is that several of these benefits generalize as well [24, 11].

We can associate to any set system $\mathcal{F}$ with ground set X some parameters inspired by convexity properties, for instance:

- The *Helly number* $\mathrm{h}_{\mathcal{F}}$ of $\mathcal{F}$ is the smallest integer h with the following property: If in a finite subfamily $\mathcal{G} \subset \mathcal{F}$, every h members of $\mathcal{G}$ intersect, then $\mathcal{G}$ has nonempty intersection. If no such h exists, we set $\mathrm{h}_{\mathcal{F}} = \infty$.

- The *Radon number* $\mathrm{r}_{\mathcal{F}}$ of $\mathcal{F}$ is the smallest integer r such that every r-element subset $S \subset X$ can be partitioned into two nonempty parts $S = P_1 \sqcup P_2$ such that $\mathrm{conv}_{\mathcal{F}}(P_1) \cap \mathrm{conv}_{\mathcal{F}}(P_2) \neq \emptyset$. (The set $\mathrm{conv}_{\mathcal{F}}(P)$, the *$\mathcal{F}$-convex hull* of a subset $P \subset X$, is the intersection of all the members of $\mathcal{F}$ that contain P.) If no such r exists, we set $\mathrm{r}_{\mathcal{F}} = \infty$.

Hence, letting $\mathcal{C}_d$ denote the set of all halfspaces in $\mathbb{R}^d$, Radon's lemma asserts that $\mathrm{r}_{\mathcal{C}_d} = d+2$ and Helly theorem that $\mathrm{h}_{\mathcal{C}_d} = d+1$. A classical result by Levi [21] asserts that for every set system $\mathcal{F}$ we have $\mathrm{h}_{\mathcal{F}} \leq \mathrm{r}_{\mathcal{F}} - 1$. (This is often stated for convexity spaces but it holds for set systems, see [2, App. C].) Similar relations between such parameters have been investigated over the years, like for instance the partition conjecture of Eckhoff [10] refuted by Bukh [6].

Conjecture 1 pertains to a parameter inspired by the *fractional Helly theorem* [20, 18], which asserts that in $(d+1)$-wise intersection hypergraphs of convex sets of $\mathbb{R}^d$, positive density implies a linear-size clique. Here is the associated parameter:

- The *fractional Helly number* $\mathrm{fh}_{\mathcal{F}}$ of $\mathcal{F}$ is the smallest integer s such that there exists a function $\beta_{\mathcal{F}} : (0,1) \to (0,1)$ with the following property: For every finite subfamily $\mathcal{F}' \subset \mathcal{F}$, whenever a fraction α of the s-tuples of $\mathcal{F}'$ have nonempty intersection, a subset $\mathcal{G}$ of $\mathcal{F}'$ of size $\beta_{\mathcal{F}}(\alpha)|\mathcal{F}'|$ has nonempty intersection.

Again, the fractional Helly theorem states that $\mathrm{fh}_{\mathcal{C}_d} = d+1$. The significance of the fractional Helly number was highlighted by Alon et al. [1], who proved that intersection-closed set systems with bounded fractional Helly number enjoy a *weak ε-net theorem*, a *Tverberg-type theorem*, a *selection lemma*, etc. It is tempting to reformulate Conjecture 1 as

> *For any function $\Psi : \mathbb{N} \to \mathbb{N}$ such that $\Psi(n) = O(n^d)$, every set set system $\mathcal{F}$ in $\mathbb{R}^d$ such that $\phi^{(d)}_{\mathcal{F}} \leq \Psi$ has fractional Helly number at most $d+1$.*

We note, however, that this statement is weaker than Conjecture 1 in that it does not assert that the function $\beta()$ underpinning the fractional Helly number depends only on Ψ and d.

1.1.2 Homological background: homological minors

A classical way to extend the theory of planar graphs is to consider embeddings of graphs into surfaces and of simplicial complexes into $\mathbb{R}^d$ or other topological spaces.

There is a rich theory of embedding of graphs on surfaces, both structural (a classic being the Heawood inequality [31]) and computational (*e.g.* the use of graph genus for parameterized complexity). Let us mention, in particular, the strong Hanani-Tutte theorem which asserts that a graph is planar if it can be drawn so that every pair of independent edges cross an even number of times. This statement generalizes to the projective plane [29, 9] but was found to fail in genus 4 [12].

Going to dimension higher than 2 changes the nature of the problems drastically, already because Fáry's theorem no longer holds. (For every $d \geq 3$ there are simplicial complexes that embed in $\mathbb{R}^d$ piecewise linearly but not linearly.) It is thus sometimes convenient to relax the notion of embedding and work with chain maps, and this was done in particular to analyze intersection patterns [14, 28, 13] using a notion of *homological minors* [35].

Formally, the *support of a singular chain* is the union of (the images of) the singular simplices with nonzero coefficient in that chain, and the *support of a simplicial chain* is the subcomplex induced by the simplices with nonzero coefficient in that chain. We write $\mathrm{supp}(\sigma)$ for the support of a (singular or simplicial) chain σ. A chain map $a : C_*(K) \to \mathrm{C}^{\mathrm{sing}}_*(X)$ (resp. $a : C_*(K) \to \mathcal{C}_*(T)$) is *nontrivial* if, for every vertex v of K, the support of $a(v)$ has odd size. Two faces in a simplicial complex K are *adjacent* if they have at least one vertex in common. A *homological almost-embedding* of a simplicial complex K into a topological space X (resp. into another simplicial complex L) is a nontrivial chain map $a : C_*(K) \to \mathrm{C}^{\mathrm{sing}}_*(X)$ (resp.

$a : C_*(K) \to C_*(L)$) such that any two non-adjacent faces $\sigma, \tau \in K$ have images with disjoint support, that is $\mathrm{supp}(a(\sigma)) \cap \mathrm{supp}(a(\tau)) = \emptyset$. In particular, for every embedding f the associated chain map $f_\#$ is a homological (almost) embedding.

Let Δ_N denote the N-dimensional simplex and for K a simplicial complex let $K^{(t)}$ denote its t-dimensional skeleton. It turns out that there is a homological version of the van Kampen-Flores theorem (see for instance [14, Corollary 14]):

▶ **Theorem 2.** *For any $d \geq 1$, $\Delta_{d+2}^{(\lceil d/2 \rceil)}$ does not homologically almost embed into $\mathbb{R}^d$.*

A simplicial complex K is a *homological minor* of a topological space or simplicial complex X if there is a homological almost-embedding of K into X. Theorem 2 thus asserts that $\Delta_{d+2}^{(\lceil d/2 \rceil)}$ is not a homological minor of $\mathbb{R}^d$, *i.e.*, that $\mathbb{R}^d$ has $\Delta_{d+2}^{(\lceil d/2 \rceil)}$ as *forbidden homological minor*.

1.1.3 Parameters of set systems with a forbidden homological minor

Matoušek [23] bounded the Helly number of topological set systems in $\mathbb{R}^d$ in which every subfamily intersects in a bounded number of connected components, all contractible. His approach starts from a set systems in $\mathbb{R}^d$, uses Ramsey theory to build a map from $\Delta_{d+2}^{(\lceil d/2 \rceil)}$ into $\mathbb{R}^d$ that is "constrained" by the known intersection patterns of $\mathcal{F}$ so that the intersection forced by the van Kampen-Flores theorem reveals a new intersection in $\mathcal{F}$.

This approach was generalized by Goaoc et al. [14] to set systems in $\mathbb{R}^d$ of bounded $\lceil d/2 \rceil$-level topological complexity, where the *h-level topological complexity* of $\mathcal{F}$ is the maximum over $\mathbb{N}$ of the homological shatter function $\phi_{\mathcal{F}}^{(h)}$, that is

$$\mathrm{hc}_{\mathcal{F}}^{(h)} := \max \left\{ \max_{0 \leq i < h} \tilde{\beta}_i \left(\bigcap_{A \in \mathcal{G}} A; \mathbb{Z}_2 \right) \;\middle|\; \mathcal{G} \subset \mathcal{F} \right\}. \tag{2}$$

That generalization relied on homological minors and replaced the construction of the "constrained map" by the (simpler) construction of a "constrained chain map". The only aspect of the method that is specific to $\mathbb{R}^d$ is the use of the forbidden homological minor given by Theorem 2, so the approach readily generalizes to set systems whose ground set is a topological space with a forbidden homological minor, see the discussions in [28, §5.2] and [13, §2.2]. This method was refined and extended to analyze other parameters of set systems whose ground set has a forbidden homological minor [28, 26, 13].

1.1.4 Proof of Conjecture 1 for bounded homological shatter functions

Conjecture 1 was proven for bounded homological shatter functions in three steps:

- First, Holmsen and Lee [17, Theorem 1.1] proved that the fractional Helly number of *any* set system can be bounded by a function of its Radon number. Specifically, letting $\Psi_{\mathrm{r}\to\mathrm{fh}}(x)$ denote the supremum of the fractional Helly number of a set system with Radon number x, they proved that for every $r \geq 3$ we have $\Psi_{\mathrm{r}\to\mathrm{fh}}(r) \leq r^{r^{\lceil \log_2 r \rceil} + r \lceil \log_2 r \rceil}$.
- Second, Patáková [28, Theorem 2.1] proved that the Radon number of *any* set system whose ground set has K as forbidden homological minor can be bounded by a function of its $(\dim K)$-level topological complexity.
- Third, Goaoc, Holmsen and Patáková [13, Theorem 1.2] proved that for every set systems $\mathcal{F}$ whose ground set has K as forbidden homological minor, if the fractional Helly number $\mathrm{fh}_{\mathcal{F}}$ is bounded then it is at most $\mu(K) + 1$, where $\mu(K)$ denotes the maximum sum of dimensions of two disjoint simplices in K.

1.2 Statement of the results

Our main contribution is to confirm Conjecture 1 for some diverging homological shatter functions and on some manifolds. This decomposes into five independent results.

Throughout the paper, we work with compact piecewise-linear (PL) manifolds (possibly with boundary); see [32, §1] for an introduction. We first generalize Theorem 2:

▶ **Theorem 3.** *For every integers $d \geq 3$ and b there exists $N = N(d, b)$ such that $\Delta_N^{(\lceil d/2 \rceil)}$ does not homologically almost embed in any compact, $(\lceil d/2 \rceil - 1)$-connected, d-dimensional PL manifold (possibly with boundary) $\mathcal{M}$ with $\beta_{\lceil d/2 \rceil}(\mathcal{M}; \mathbb{Z}_2) \leq b$.*

This partially answers [28, Problem 3], [13, Conjecture 1.7] and [26, Conjecture 2] and extends the previous confirmation of Conjecture 1 from set systems in $\mathbb{R}^d$ to set systems on manifolds. This also extends several results on set systems with bounded topological complexity (Helly's theorem, Radon's theorem, (p, q)-theorem, ...) from $\mathbb{R}^d$ to sufficiently connected manifolds (see [28, 13]). We can relax the connectivity assumption (see [2, App. B]) but not remove it.

One ingredient in the proof of Theorem 3 is the following analogue of the Hanani-Tutte theorem for homological almost-embeddings, which is of independent interest.

▶ **Theorem 4.** *Let K be a simplicial complex of dimension $k > 1$ and let $\mathcal{M}$ be a compact $2k$-dimensional PL manifold (possibly with boundary). If there exists a triangulation T of $\mathcal{M}$ and a non-trivial chain map $f : C_*(K; \mathbb{Z}_2) \to C_*(T; \mathbb{Z}_2)$ in general position such that the images of any two non-adjacent k-faces $\sigma, \tau \in K$ intersect in an even number of points, then K is a homological minor of $\mathcal{M}$.*

We formalize what we mean by general position in Section 2.

The homological shatter function $\Phi_{\mathcal{F}}^{(h)}$ defined in Equation (1) can be reformulated as a *graded* version of the topological complexity $\mathrm{hc}_{\mathcal{F}}^{(h)}$ defined in Equation (2), where the value for parameter t considers only intersections of subfamilies of size at most t: $\phi_{\mathcal{F}}^{(h)}(t) = \sup_{\substack{\mathcal{F}' \subset \mathcal{F} \\ |\mathcal{F}'| \leq t}} \mathrm{hc}_{\mathcal{F}'}^{(h)}$. We systematize this viewpoint and define graded analogues of other parameters of set systems. For instance here are the *graded Radon* and *graded Helly* numbers:

$$\mathrm{r}_{\mathcal{F}}(t) \coloneqq \sup_{\substack{\mathcal{F}' \subset \mathcal{F} \\ |\mathcal{F}'| \leq t}} \mathrm{r}_{\mathcal{F}'} \quad \text{and} \quad \mathrm{h}_{\mathcal{F}}(t) \coloneqq \sup_{\substack{\mathcal{F}' \subset \mathcal{F} \\ |\mathcal{F}'| \leq t}} \mathrm{h}_{\mathcal{F}'}. \tag{3}$$

It is straightforward to see that if the graded Helly numbers of a set system do not grow fast enough, then they are ultimately stationary and the set system has bounded Helly number (Lemma 10). We prove a similar condition for (graded) Radon numbers:

▶ **Theorem 5.** *Let $\mathcal{F}$ be a set system. If $\lim_{t \to \infty} \mathrm{r}_{\mathcal{F}}(t) - \log_2 t = -\infty$, then $\mathrm{r}_{\mathcal{F}} < \infty$.*

As a application, we extend Patáková's theorem from bounded to sufficiently slowly diverging homological shatter function (Corollary 12). This, in turns yields fractional Helly theorems for sufficiently slowly diverging homological shatter functions.

▶ **Corollary 6.** *For every simplicial complex K there exists a function $\Psi_K : \mathbb{N} \to \mathbb{N}$ with $\lim_{t \to \infty} \Psi_K(t) = +\infty$ such that the following holds. If $\mathcal{F}$ is a set system whose ground set has K as forbidden homological minor and such that $\phi_{\mathcal{F}}^{(\dim K)}(t) \leq \Psi_K(t)$ for t large enough, then $\mathcal{F}$ has fractional Helly number at most $(\mu(K) + 1)$.*

We then investigate more graded numbers and establish more relations between graded and ungraded numbers. As an application, we extend the Holmsen-Lee bound on $\Psi_{\mathrm{r} \to \mathrm{fh}}(\cdot)$. Let $\Xi : \mathbb{N} \to \mathbb{N}$ be the function defined by $\Xi(r) := r^{r^{\lceil \log_2 r \rceil} + r \lceil \log_2 r \rceil}$.

▶ **Theorem 7.** *Let $\Psi : \mathbb{N} \to \mathbb{N}$ and $t_0 \in \mathbb{N}$ such that $\Psi(t) < t + 1$ for every $t \geq t_0$. If there exists an integer $t_1 \geq {t_0}^2$ such that $\Xi(\Psi(t_1)) < \frac{t_1}{t_0}$, then every set system whose graded Radon number function is bounded from above by Ψ has bounded fractional Helly number.*

With Theorem 7 we can strengthen Corollary 6, as a close inspection of the proof reveals that the function β associated to the fractional Helly number depends only on Ψ and t_0. (It also allows a slightly faster growth than Corollary 12.) We postpone this to the full version of the paper.

2 On homological minors

In this section we prove Theorems 3 and 4. For completeness, we start with a consequence of the simplicial approximation theorem that allows us to work purely in simplicial homology:

▶ **Lemma 8** ([2, App. A])**.** *A simplicial complex K is a homological minor of a compact PL manifold (possibly with boundary) $\mathcal{M}$ if and only if K is a homological minor of some triangulation of $\mathcal{M}$.*

When counting intersection points between chains, we focus on intersections that are stable under small perturbation. We therefore consider generic intersections, meaning intuitively that the chains intersect transversally. We formalize this in terms of linking numbers.[1]

Let X be a triangulation of $\mathbb{S}^{2k-1}$. Let S_1 and S_2 be two subcomplexes of X with $|S_1|$ and $|S_2|$ homeomorphic to $\mathbb{S}^{k-1}$. Suppose that the simplicial complex $X \setminus S_2$ induced by X on the vertices not in S_2 has a geometric realization homotopy equivalent to $\mathbb{S}^{2k-1} \setminus \mathbb{S}^{k-1}$ (this can always be ensured up to taking a subdivision of X). The *linking number* of S_1 and S_2 in X is 1 if the homology class $[S_1]$ generates $H_{k-1}(X \setminus S_2; \mathbb{Z}_2) \cong \mathbb{Z}_2$, and 0 if $[S_1]$ is trivial in $H_{k-1}(X \setminus S_2; \mathbb{Z}_2)$. (Exchanging S_1 and S_2 in this definition yields the same result.)

Let T be a triangulation of a compact PL manifold (possibly with boundary) $\mathcal{M}$. Two k-chains $z_1, z_2 \in C_k(T; \mathbb{Z}_2)$ *intersect generically* in vertex v if the closed star B of v in T satisfies: $B \cap z_1 \cap z_2 = \{v\}$, $D_1 := z_1 \cap B$ and $D_2 := z_2 \cap B$ are k-dimensional balls, and the spheres ∂D_1 and ∂D_2 have linking number 1 in ∂B. Two k-chains $z_1, z_2 \in C_k(T; \mathbb{Z}_2)$ are *in general position* if $z_1 \cap z_2$ consists of finitely many vertices, and z_1 and z_2 intersect generically in each of these vertices. Two k-chains $z_1, z_2 \in C_k(T; \mathbb{Z}_2)$ *intersect evenly* if they are in general position and $\mathrm{im}(z_1) \cap \mathrm{im}(z_2)$ has even size.

For K a k-dimensional complex, a simplicial chain map $f : C_\bullet(K; \mathbb{Z}_2) \to C_\bullet(T; \mathbb{Z}_2)$ is in *general position* if for every non-adjacent k-faces $\sigma, \tau \in K$, the chains $f(\sigma)$ and $f(\tau)$ are in general position and for each vertex $v \in f(\sigma) \cap f(\tau)$, σ and τ are the only faces of K whose images under f intersect the closed star of v in T. We say that a simplicial map $f : K \to T$ is in *general position* if the associated chain map $f_\#$ is.

For any compact PL manifold (possibly with boundary) $\mathcal{M}$ of dimension $2k$, there exists a map $\cap_\mathcal{M} : H_k(\mathcal{M}; \mathbb{Z}_2) \times H_k(\mathcal{M}; \mathbb{Z}_2) \to \mathbb{Z}_2$, called the *intersection form* of $\mathcal{M}$, such that $\cap_\mathcal{M}([z_1], [z_2]) \in \mathbb{Z}_2$ counts the intersection points of z_1 and z_2 modulo 2. We use no property of intersection forms besides their existence, and refer the interested reader to Prasolov [30, Chapter 2, §2.7] for a precise definition and to Paták and Tancer [27] for an brief account.

The next result is due to Paták and Tancer [27, Proposition 21] and formulated following the presentation of Skopenkov [33], which is better suited for our purpose. (More precisely, we reformulated [33, Theorem 1.1.5] using [33, Lemma 2.1.1].) Note that the proofs by Paták-Tancer [27] and by Skopenkov [33] use only PL maps.

[1] Intuitively, this generalizes the idea that in the plane, two curves cross at a point x if the branches of the curves alternate around x.

▶ **Theorem 9.** *Let L be a simplicial complex of dimension $k > 1$ and let $\mathcal{M}$ be a compact, $(k-1)$-connected, $2k$-dimensional PL manifold (possibly with boundary). The following statements are equivalent:*

- **(i)** *There exists a triangulation T of $\mathcal{M}$ and a simplicial map $f : L \to T$ in general position such that the images of any two non-adjacent faces intersect evenly.*
- **(ii)** *There exists a triangulation R of $\mathbb{R}^{2k}$ and a simplicial map $g : L \to R$ in general position and a map α that sends each k-face of L to an element of $H_k(\mathcal{M}; \mathbb{Z}_2)$ such that any two non-adjacent k-faces $\sigma, \tau \in L$ have images that intersect in an even number of points if and only if $\cap_{\mathcal{M}}(\alpha(\sigma), \alpha(\tau)) = 0$.*

2.1 A homological Hanani-Tutte theorem

Let us now prove Theorem 4. Let K be a simplicial complex of dimension $k > 1$ and let T be a triangulation of a compact PL manifold (possibly with boundary) $\mathcal{M}$ of dimension $2k$. Let $f : C_*(K; \mathbb{Z}_2) \to C_*(T; \mathbb{Z}_2)$ be a non-trivial chain map in general position such that the images of any two non-adjacent k-faces $\sigma, \tau \in K$ intersect evenly.

Our goal is to prove that K is a homological minor of $\mathcal{M}$. This requires repeatedly subdividing the triangulation T. In what follows, every time we subdivide a triangulation T_i into T_{i+1}, all chain maps to T_i and subcomplexes of T_i are also subdivided to T_{i+1}. Also, throughout the proof we identify every pure ℓ-dimensional simplicial complex with the unique ℓ-chain with coefficients in $\mathbb{Z}_2$ it supports; we abuse the terminology and say that we add (pure) simplicial complexes over $\mathbb{Z}_2$ to mean that we add the corresponding chains.

If f is a homological almost-embedding we are done. Otherwise, there exist some non-adjacent k-faces $\sigma, \tau \in K$ such that $f(\sigma) \cap f(\tau)$ is non-empty. By assumption, this intersection is a set of vertices of even size, so let $x, y \in f(\sigma) \cap f(\tau)$ be two distinct such vertices. Recall that f is in general position, $f(\sigma)$ and $f(\tau)$ intersect generically in x and in y. We set out to construct a refinement T' of T and a new map $f' : C_*(K; \mathbb{Z}_2) \to C_*(T'; \mathbb{Z}_2)$ that is also in general position, differs from f only on σ and τ, and satisfies $f'(\sigma) \cap f'(\tau) = (f(\sigma) \cap f(\tau)) \setminus \{x, y\}$ as well as $f'(\alpha) \cap f'(\beta) = f(\alpha) \cap f(\beta)$ for any $\alpha \in \{\sigma, \tau\}$ and any k-face $\beta \in K \setminus \{\sigma, \tau\}$. Iterating this procedure produces the announced homological almost-embedding of K into a triangulation of $\mathcal{M}$.



Figure 1 The setup for the construction of T' and f'.

Let B_x and B_y denote the closed stars of x and y in T. For $z \in \{x, y\}$ and $\alpha \in \{\sigma, \tau\}$ let $D_{\alpha,z} := \operatorname{supp}(f(\alpha)) \cap B_z$. Since $f(\sigma)$ and $f(\tau)$ intersect generically in x and y, the complexes $D_{\sigma,x}, D_{\sigma,y}, D_{\tau,x}$ and $D_{\tau,y}$ are k-dimensional balls, the $(k-1)$-spheres $\partial D_{\sigma,x}$ and $\partial D_{\tau,x}$ have linking number 1 in ∂B_x, and similarly $\partial D_{\sigma,y}$ and $\partial D_{\tau,y}$ have linking number 1 in ∂B_y.

We subdivide T into T_1 so that the closed star B'_x of x and B'_y of y in T_1 are disjoint from ∂B_x and ∂B_y, respectively. In particular, for $z \in \{x, y\}$ and $\alpha \in \{\sigma, \tau\}$, letting $D'_{\alpha,z} := \operatorname{supp}(f(\alpha)) \cap B'_z$, the pair $(\partial D'_{\alpha,z}, \partial B'_z)$ is homeomorphic to $(\mathbb{S}^{k-1}, \mathbb{S}^{2k-1})$. The

genericity of x and y in $f(\sigma) \cap f(\tau)$ is preserved through the subdivision $T \to T_1$ so the four complexes $D'_{\bullet,\bullet}$ are k-dimensional balls and their bounding spheres have the same linking numbers in $\partial B'_x$ and $\partial B'_y$ as their counterparts on ∂B_x and ∂B_y.

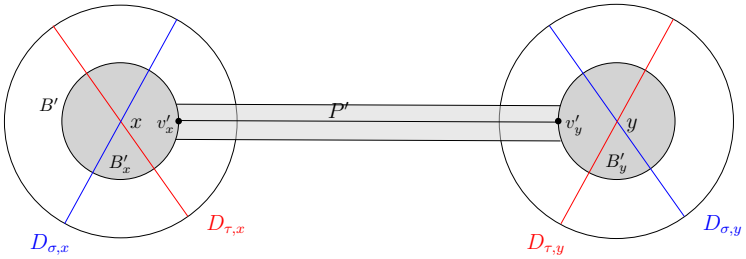


Figure 2 The construction of B'.

Up to subdividing T_1 into T_2, there exist vertices v'_x and v'_y in $\partial B'_x$ and $\partial B'_y$, respectively, and a path P' in the 1-skeleton of T_2 from v'_x to v'_y such that, letting $N(P')$ denote the closed star of P' in T_2, the closed star of $N(P')$ is disjoint from the image of f. The union $B' := B'_x \cup N(P') \cup B'_y$ is a ball. Observe that in $\partial B'$, the $(k-1)$-spheres $\partial D'_{\sigma,x}$ and $\partial D'_{\tau,x}$ retain the linking number 1 that they have on $\partial B'_x$. Similarly, in $\partial B'$, the $(k-1)$-spheres $\partial D'_{\sigma,y}$ and $\partial D'_{\tau,y}$ retain the linking number 1 that they have on $\partial B'_y$.

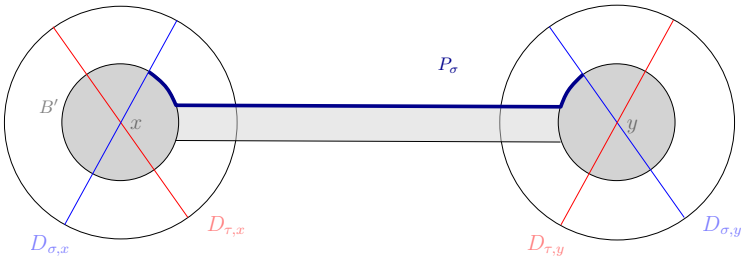


Figure 3 The path P_σ.

Thus, up to further subdividing T_2 into T_3, there exist a path P_σ in the 1-skeleton of $\partial B'$ that connects $\partial D'_{\sigma,x}$ to $\partial D'_{\sigma,y}$ and with relative interiors disjoint from the image of f. (Here we use that $\partial D'_{\alpha,z}$ is of codimension $k \geq 2$ in $\partial B'$.) Up to subdividing the triangulation further, we can pipe $D_{\sigma,x}$ and $D_{\sigma,y}$ together by a tube F_σ found in a neighborhood of the path P_σ [32, §5.10]. The tube $(F_\sigma, F_\sigma \cap D_{\sigma,x}, F_\sigma \cap D_{\sigma,y})$ is homeomorphic to $\big(\mathbb{D}^k \times [0,1], \mathbb{D}^k \times \{0\}, \mathbb{D}^k \times \{1\}\big)$. Moreover, by the (PL) general position theorem for embeddings [32, §5.3], we can take F_σ such that $\partial B'$ and ∂F_σ intersect in a generic way. By taking F_σ in a sufficiently small neighborhood U of P_σ so that $(U, \partial B' \cap U)$ is homeomorphic to $(\mathbb{R}^{2k}, \mathbb{R}^{2k-1} \times \{0\})$, and the triple $(F_\sigma \cap \partial B', F_\sigma \cap D_{\sigma,x} \cap \partial B', F_\sigma \cap D_{\sigma,y} \cap \partial B')$ is homeomorphic to $\big(\mathbb{D}^{k-1} \times [0,1], \mathbb{D}^{k-1} \times \{0\}, \mathbb{D}^{k-1} \times \{1\}\big)$.

Now, let C_σ be the sum $D_{\sigma,x} + \partial F_\sigma + D_{\sigma,y}$ (over $\mathbb{Z}_2$). Note that C_σ is contained in the union of $B_x \cup B_y$ and the closed star of $N(P')$, and therefore intersects the image of f only inside $B_x \cup B_y$. We note that $F_\sigma \cap \partial B'$ also pipes the $(k-1)$-spheres $D_{\sigma,x} \cap \partial B'$ and $D_{\sigma,y} \cap \partial B'$. Indeed, $(F_\sigma \cap \partial B', F_\sigma \cap D_{\sigma,x} \cap \partial B', F_\sigma \cap D_{\sigma,y} \cap \partial B') = (F_\sigma \cap \partial B', F_\sigma \cap \partial(D_{\sigma,x} \cap B'), F_\sigma \cap \partial(D_{\sigma,y} \cap B'))$ is homeomorphic to $\big(\mathbb{D}^{k-1} \times [0,1], \mathbb{D}^{k-1} \times \{0\}, \mathbb{D}^{k-1} \times \{1\}\big)$. It follows that $C_\sigma \cap \partial B'$ is homeomorphic to the connected sum of two $(k-1)$-spheres, and is therefore homeomorphic to a $(k-1)$-sphere.

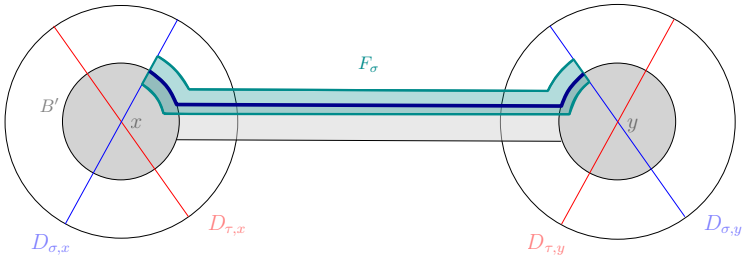


Figure 4 The piping F_σ between $D_{\sigma,x}$ and $D_{\sigma,y}$ in a neighborhood of P_σ.

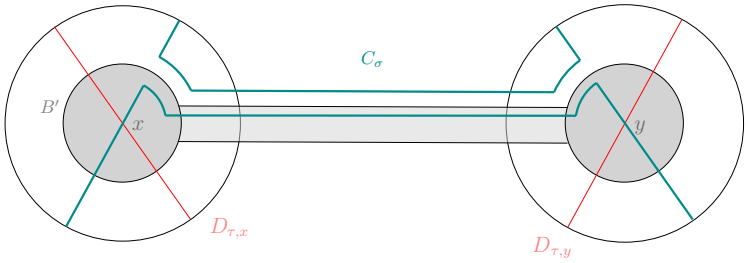


Figure 5 The chain $C_\sigma = D_{\sigma,x} + \partial F_\sigma + D_{\sigma,y}$ over $\mathbb{Z}_2$.

We claim that, in $\partial B'$, the linking number between $C_\sigma \cap \partial B'$ and $D_{\tau,x} \cap \partial B'$ equals the linking number between $D_{\sigma,x} \cap \partial B'$ and $D_{\tau,x} \cap \partial B'$, that is 1. This follows from the fact that the chain C_σ differs from $D_{\sigma,x}$ by $\partial F_\sigma + D_{\sigma,y}$, and that each of $\partial F_\sigma \cap \partial B'$ and $D_{\sigma,y} \cap \partial B'$ is a boundary in $\partial B' \setminus D_{\tau,x}$. The same claim holds if we exchange x for y.

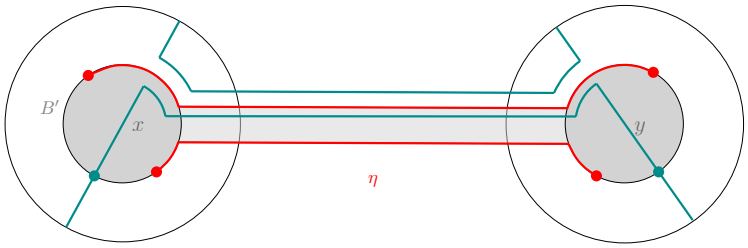


Figure 6 The chain $\eta \in C_k(\partial B' \setminus C_\sigma; \mathbb{Z}_2)$ such that $\partial\eta = D_{\tau,x} \cap \partial B' + D_{\tau,y} \cap \partial B'$ over $\mathbb{Z}_2$.

Altogether, we get that $D_{\tau,x} \cap \partial B'$ and $D_{\tau,y} \cap \partial B'$ are in the same homology class in $\partial B' \setminus C_\sigma$. Hence, their sum is a boundary, and there exists a chain $\eta \in C_k(\partial B' \setminus C_\sigma; \mathbb{Z}_2)$ such that the support of $\partial\eta$ is the sum over $\mathbb{Z}_2$ of $D_{\tau,x} \cap \partial B'$ and $D_{\tau,y} \cap \partial B'$. We finally set

$$f'(\sigma) = \underbrace{f(\sigma) + D_{\sigma,x} + D_{\sigma,y}}_{\substack{f(\sigma) \text{ with its restriction to } B_x \cup B_y \text{ removed}}} + C_\sigma \qquad \text{and} \qquad f'(\tau) = \underbrace{f(\tau) + D'_{\tau,x} + D'_{\tau,y}}_{\substack{f(\tau) \text{ with its restriction to } B' \text{ removed}}} + \eta$$

We set $f'(\omega) = f(\omega)$ for every other face $\omega \in K$. Extending f' linearly yields a chain map, since both $f'(\sigma) - f(\sigma)$ and $f'(\tau) - f(\tau)$ are cycles [2]. The chain map f' is as announced: $f'(\sigma) \cap f'(\tau) = f(\sigma) \cap f(\tau) \setminus \{x, y\}$ and every other intersection remains unchanged. This concludes the proof of Theorem 4.

[2] They are even boundaries, ensuring that f' is chain homotopic to f

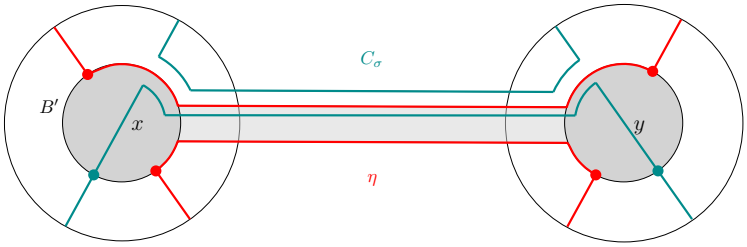


Figure 7 The chains C_σ and η reroute $f(\sigma)$ and $f(\tau)$ so as to remove intersections in B_x and B_y.

2.2 Forbidden homological minors for manifolds

We now prove Theorem 3. First, note that the odd-dimensional case reduces to the even-dimensional one. Indeed, for every $k \geq 2$, if $\mathcal{M}$ is a compact, $(k-1)$-connected, $(2k-1)$-dimensional PL manifold (possibly with boundary), then $\mathcal{M} \times [0,1]$ is a compact, $(k-1)$-connected, $2k$-dimensional PL manifold (possibly with boundary). Moreover, $\beta_k(\mathcal{M} \times [0,1]; \mathbb{Z}_2) = \beta_k(\mathcal{M}; \mathbb{Z}_2)$ and any homological minor of $\mathcal{M}$ is a homological minor of $\mathcal{M} \times [0,1]$. If the statement holds for d even and $b \in \mathbb{N}$ with some $N(b,d)$, then it holds with $d-1$ and b by putting $N(d-1,b) := N(d,b)$. So let us now consider the even-dimensional case.

Let $k \geq 2$ and $b \in \mathbb{N}$, and let $\mathcal{M}$ be a compact, $(k-1)$-connected, $2k$-dimensional PL manifold (possibly with boundary). Let us fix N and suppose that $K = \Delta_N^{(k)}$ is a homological minor of $\mathcal{M}$. By Lemma 8, there exist a triangulation T of $\mathcal{M}$ and a (simplicial) homological almost-embedding $C_*(K) \to C_*(T)$. Actually, letting $S := T^{(k)}$, we have that there exists a homological almost-embedding $a : C_*(K; \mathbb{Z}_2) \to C_*(S; \mathbb{Z}_2)$.

We apply Theorem 9 with $L = S$. Condition (i) holds with $T = S$ and f the identity, so Condition (ii) also holds. Hence, there exists a triangulation R of $\mathbb{R}^{2k}$, a simplicial map $g : S \to R$ in general position, and a map α that sends each k-face of S to an element of $H_k(\mathcal{M}; \mathbb{Z}_2)$ such that any two non-adjacent k-faces $\sigma, \tau \in S$ intersect evenly if and only if $\cap_{\mathcal{M}}(\alpha(\sigma), \alpha(\tau)) = 0$. We let $\tilde{\alpha} : C_k(S) \to H_k(\mathcal{M}; \mathbb{Z}_2)$ denote the linear extension of α.

Consider the chain map $b : C_*(K; \mathbb{Z}_2) \to C_*(R; \mathbb{Z}_2)$ defined by $b = g_\# \circ a$. Note that b is nontrivial since a is nontrivial and g is a simplicial map. Moreover, the fact that b is a chain map in general position follows from three observations:

- since g is a simplicial map in general position, $g_\#$ is a chain map in general position,
- since a is a homological almost-embedding, it is also a chain map in general position, and
- the composition of a homological almost-embedding and a chain map in general position is a chain map in general position.

Notice that for $N \geq 2k+3$, not every independent k-faces of K can have images under b that intersect evenly. Indeed Theorem 4 would then imply that K is a homological minor of $\mathbb{R}^{2k}$, which would contradict the homological van Kampen-Flores theorem (Theorem 2). We use Ramsey's theorem to show that this contradiction can be reached for some subcomplex of K.

So consider two non-adjacent k-faces $\sigma, \tau \in K$ and put $a(\sigma) = \sigma_1 + \sigma_2 + \ldots + \sigma_s$ and $a(\tau) = \tau_1 + \tau_2 + \ldots + \tau_t$. Since a is a homological almost-embedding, $\mathrm{supp}(a(\sigma))$ and $\mathrm{supp}(a(\tau))$ are disjoint, and σ_i and τ_j are thus non-adjacent for every $(i,j) \in [s] \times [t]$. Hence, given $(i,j) \in [s] \times [t]$, $g(\sigma_i)$ and $g(\tau_j)$ intersect evenly if and only if $\cap_{\mathcal{M}}(\alpha(\sigma_i), \alpha(\tau_j)) = 0$. We can thus count the intersections of $b(\sigma)$ and $b(\tau)$ (the following equalities are modulo 2 and Card () denotes the cardinal):

$$\begin{aligned}\operatorname{Card}(b(\sigma)\cap b(\tau)) &= \sum_{i\in[s],j\in[t]} \operatorname{Card}(g(\sigma_i)\cap g(\tau_j))\\ &= \sum_{i\in[s],j\in[t]} \cap_{\mathcal{M}}(\alpha(\sigma_i),\alpha(\tau_j))\\ &= \sum_{i\in[s]} \cap_{\mathcal{M}}(\alpha(\sigma_i),\tilde{\alpha}(a(\tau))) = \cap_{\mathcal{M}}(\tilde{\alpha}(a(\sigma)),\tilde{\alpha}(a(\tau))).\end{aligned}$$

Let $r = \beta_k(\mathcal{M};\mathbb{Z}_2)$. The map $\beta : C_k(K) \to H_k(\mathcal{M};\mathbb{Z}_2)$ defined by $\beta = \tilde{\alpha}\circ a$ induces a coloring of the k-simplices of K by the (at most 2^r) elements of $H_k(M;\mathbb{Z}_2)$. Let N' denote the number of vertices of $\operatorname{sd}\Delta^{(k)}_{2k+2}$. By the hypergraph Ramsey theorem, for N large enough (as a function of r and k) there exists a subset W of N' vertices in K such that β is constant over all k-simplices of $K[W]$; let us denote by $\square$ this constant value. In particular, for every k-faces σ,τ of $K[W]$ such that σ and τ are not adjacent, we have, modulo 2, $\operatorname{Card}(b(\sigma)\cap b(\tau)) = \cap_{\mathcal{M}}(\beta(\sigma_1),\beta(\tau_1)) = \cap_{\mathcal{M}}(\square,\square)$. Let us fix a bijection from the vertices of $\operatorname{sd}\Delta^{(k)}_{2k+2}$ to W and extend it to a chain map $j : C_*\left(\operatorname{sd}\Delta^{(k)}_{2k+2}\right) \to C_*(K[W])$. Also, let $h : C_*\left(\Delta^{(k)}_{2k+2}\right) \to C_*\left(\operatorname{sd}\Delta^{(k)}_{2k+2}\right)$ denote the subdivision chain map, where each i-face of $\Delta^{(k)}_{2k+2}$ is mapped to the sum of the i-faces of $\operatorname{sd}\Delta^{(k)}_{2k+2}$ that it contains. In particular, for every k-face σ of $\Delta^{(k)}_{2k+2}$, the chain $h(\sigma)$ is supported on $k!$ k-faces of $\operatorname{sd}\Delta^{(k)}_{2k+2}$.

Let us examine the properties of $b\circ j\circ h$. First, it is a nontrivial chain map (because b, j and h are). Moreover, $j\circ h$ is a homological almost-embedding (since j and h are), and its composition with the chain map in general position b yields a chain map in general position. Furthermore, any two non-adjacent k-faces $\sigma,\tau \in \Delta^{(k)}_{2k+2}$ have images under $b\circ j\circ h$ that intersect evenly. To see this, let us put $j\circ h(\sigma) = \sigma_1+\sigma_2+\ldots+\sigma_s$ and $j\circ h(\tau) = \tau_1+\tau_2+\ldots+\tau_s$. Since $j\circ h$ is a homological almost-embedding, $\operatorname{supp}(j\circ h(\sigma))$ and $\operatorname{supp}(j\circ h(\tau))$ are disjoint. It follows that for every $i,j\in[s]$ the k-simplices σ_i and τ_j are non-adjacent. We therefore have, modulo 2,

$$\begin{aligned}\operatorname{Card}(b\circ j\circ h(\sigma)\cap b\circ j\circ h(\tau)) &= \sum_{i,j\in[s]} \operatorname{Card}(b(\sigma_i)\cap b(\tau_j))\\ &= \sum_{i,j\in[s]} \cap_{\mathcal{M}}(\beta(\sigma_i),\beta(\tau_j)) = \sum_{i,j\in[s]} \cap_{\mathcal{M}}(\square,\square).\end{aligned}$$

To sum up, the cardinal of $b\circ j\circ h(\sigma)\cap b\circ j\circ h(\tau)$ has the same parity as $\cap_{\mathcal{M}}(\square,\square)s^2$. Since $s = k!$ is even, σ and τ have images under $b\circ j\circ h$ that intersect evenly.

To conclude, $b\circ j\circ h$ is a nontrivial chain map in general position from $\Delta^{(k)}_{2k+2}$ to R such that independent faces have images that intersect evenly. By Theorem 4 this means that $\Delta^{(k)}_{2k+2}$ is a homological minor of $\mathbb{R}^{2k}$, a contradiction with Theorem 2. Thus, for N large enough, the initial hypothesis that $\Delta^{(k)}_N$ is a homological minor of $\mathcal{M}$ cannot be true.

3 Graded parameters of set systems

In this section we present our contributions on set systems of parameters.

3.1 Graded Radon and Helly numbers

Each relation between parameters of a set system yields a relation between their graded analogues. From Levi's inequality we get the following inequality between the graded Helly and Radon numbers (defined in Section 1.1.1):

$$\forall t \in \mathbb{N}, \qquad \mathrm{h}_{\mathcal{F}}(t) = \sup_{\substack{\mathcal{F}' \subset \mathcal{F} \\ |\mathcal{F}'| \leq t}} \mathrm{h}_{\mathcal{F}'} \leq \sup_{\substack{\mathcal{F}' \subset \mathcal{F} \\ |\mathcal{F}'| \leq t}} (\mathrm{r}_{\mathcal{F}'} - 1) = \mathrm{r}_{\mathcal{F}}(t) - 1. \tag{4}$$

It follows from the definitions of graded parameters that each one is a non-decreasing function that converges to the ungraded parameter (possibly ∞). We notice that if a graded Helly number is asymptotically sublinear then it is bounded:

▶ **Lemma 10.** *Let $\mathcal{F}$ be a set system and $t_0 \in \mathbb{N}$. If $\mathrm{h}_{\mathcal{F}}(t) < t$ for all $t > t_0$, then $\mathrm{h}_{\mathcal{F}} \leq t_0$.*

Proof. By definition, we have $\mathrm{h}_{\mathcal{F}}(t) \leq t$ for every $t \in \mathbb{N}$. Moreover, $\mathrm{h}_{\mathcal{F}}(t) \neq \mathrm{h}_{\mathcal{F}}(t-1)$ if and only if $\mathrm{h}_{\mathcal{F}}(t) = t$. The assumption and a straightforward induction therefore implies that for every $t > t_0$ we have $\mathrm{h}_{\mathcal{F}}(t) = \mathrm{h}_{\mathcal{F}}(t_0) \leq t_0$. ◀

The growth of graded Radon numbers is at most linear [2, App. D] and rather constrained:

▶ **Lemma 11.** *Let $\mathcal{F}$ be a set system and $t \geq 2$ an integer. If $\mathrm{r}_{\mathcal{F}}(t) > \mathrm{r}_{\mathcal{F}}(t-1)$, then $\mathrm{r}_{\mathcal{F}}(t-1) \geq 1 + \log_2\left(1 + \frac{t}{\mathrm{h}_{\mathcal{F}}(t)}\right)$.*

Proof. Let X denote the ground set of $\mathcal{F}$ and let $n = \mathrm{r}_{\mathcal{F}}(t-1)$. Suppose that $\mathrm{r}_{\mathcal{F}}(t) > n$, so that there exist a subset $\mathcal{G} = \{G_1, G_2, \ldots, G_t\} \subset \mathcal{F}$ and a subset $S \subseteq X$ of size n such that

- **(i)** there is no partition of S into two parts whose $\mathcal{G}$-convex hulls intersect,
- **(ii)** for every $i \in [t]$, there exists a partition $\mathcal{P}_i$ of S into two parts whose $(\mathcal{G} \setminus \{G_i\})$-convex hulls intersect.

Recall that given $\mathcal{F}' \subseteq \mathcal{F}$, the $\mathcal{F}'$-convex hull $\mathrm{conv}_{\mathcal{F}'}(P)$ of a subset $P \subset X$ is the intersection of all the members of $\mathcal{F}'$ that contain P. In particular, for any $P \subseteq X$ such that $P \not\subseteq G_i$ we have $\mathrm{conv}_{\mathcal{G}}(P) = \mathrm{conv}_{\mathcal{G}\setminus\{G_i\}}(P)$. Conditions (i) and (ii) therefore imply that every $G_i \in \mathcal{G}$ contains one or the other part of $\mathcal{P}_i$.

There are at most $2^{n-1} - 1$ partitions of S in two nonempty parts. Let us assume that $t > (2^{n-1}-1)h$ for some integer h, so that by the pigeonhole principle there exist $h+1$ indices $i_1, i_2, \ldots, i_{h+1}$ such that the partitions $\mathcal{P}_{i_1}, \mathcal{P}_{i_2}, \ldots, \mathcal{P}_{i_{h+1}}$ coincide. Let $\{P_1, P_2\}$ be that partition of S. Let us put $\mathcal{G}' = \{A \in \mathcal{G} \colon P_1 \subseteq A \text{ or } P_2 \subseteq A\}$. We make two observations:

- $\cap_{A \in \mathcal{G}'} A$ coincides with $\mathrm{conv}_{\mathcal{G}}(P_1) \cap \mathrm{conv}_{\mathcal{G}}(P_2)$ and is therefore empty.
- every choice of h elements in $\mathcal{G}'$ has nonempty intersection. Indeed, $\mathcal{G}'$ contains G_{i_1}, G_{i_2}, $\ldots$, $G_{i_{h+1}}$, so that any choice of h elements from $\mathcal{G}'$ is bound to miss G_{i_j} for at least one $j \in [h+1]$ and their intersection must then contain $\mathrm{conv}_{\mathcal{G}\setminus\{G_{i_j}\}}(P_1) \cap \mathrm{conv}_{\mathcal{G}\setminus\{G_{i_j}\}}(P_2) \neq \emptyset$.

For $h \geq \mathrm{h}_{\mathcal{F}}(t)$ these conditions are incompatible. We therefore have $t \leq (2^{n-1} - 1)\,\mathrm{h}_{\mathcal{F}}(t)$, and the statement follows. ◀

We can now prove that any set system with sufficiently slowly growing graded Radon numbers has finite Radon number.

Proof of Theorem 5. Let $\mathcal{F}$ be a set system with infinite Radon number. If the Helly number $\mathrm{h}_{\mathcal{F}}$ is also infinite, then by Lemma 10 there exists an increasing sequence $\{t_i\}_{i\in\mathbb{N}}$ such that $\mathrm{h}_{\mathcal{F}}(t_i) = t_i$, and therefore $\mathrm{r}_{\mathcal{F}}(t_i) \geq t_i + 1$ by Inequality (4); this prevents $\mathrm{r}_{\mathcal{F}}(t) - \log_2 t$ from going to $-\infty$ as $t \to \infty$. So suppose that the Helly number $\mathrm{h}_{\mathcal{F}}$ is finite. The assumption that $\mathrm{r}_{\mathcal{F}} = \infty$ ensures that there exists an increasing sequence $\{t_i\}_{i\in\mathbb{N}}$ such that $\mathrm{r}_{\mathcal{F}}(t_i) > \mathrm{r}_{\mathcal{F}}(t_i - 1)$. Lemma 11 implies that $\mathrm{r}_{\mathcal{F}}(t_i) > \mathrm{r}_{\mathcal{F}}(t_i - 1) \geq \log_2 t_i - \log_2 \mathrm{h}_{\mathcal{F}}$. Again, this prevents $\mathrm{r}_{\mathcal{F}}(t) - \log_2 t$ from going to $-\infty$ as $t \to \infty$. The statement follows by contraposition. ◀

3.2 Consequences for topological set systems

Let us finally consider topological set systems with slowly growing homological shatter function and ground set with a forbidden homological minor.

▶ **Corollary 12.** *For every simplicial complex K there exists a function $S_K : \mathbb{N} \to \mathbb{N}$ with $\lim_{t\to\infty} S_K(t) = +\infty$ such that the following holds. Any set system $\mathcal{F}$ whose ground set has K as forbidden homological minor and satisfies $\phi_{\mathcal{F}}^{(\dim K)}(t) \leq S_K(t)$ for t large enough has finite Radon number.*

Proof. Recall that $\Psi^{(K)}_{\mathrm{hc}\to\mathrm{r}}(x)$ denotes the supremum of the Radon number of a set system with $(\dim K)$-level topological complexity at most x and whose ground set has K as forbidden homological minor. Patáková [28] proved that $\Psi^{(K)}_{\mathrm{hc}\to\mathrm{r}}(x)$ is finite for every K and x.

For $t \in \mathbb{N}$, we define $S_K(t) = \max\{x \in \mathbb{N} \mid \Psi^{(K)}_{\mathrm{hc}\to\mathrm{r}}(x) \leq \frac{1}{2}\log_2 t\}$. This ensures that $\Psi^{(K)}_{\mathrm{hc}\to\mathrm{r}}(S_K(t)) \leq \frac{1}{2}\log_2 t$ for every $t \in \mathbb{N}$. Observe that $\lim_{t\to\infty} S_K(t) = \infty$ since $\Psi^{(K)}_{\mathrm{hc}\to\mathrm{r}}(x)$ is finite for every $x \in \mathbb{N}$.

Now consider a set system $\mathcal{F}$ with function $\phi_{\mathcal{F}}^{(\dim K)} \leq S_K$ and whose ground set has K as forbidden homological minor. Let $t \in \mathbb{N}$ and consider a subset $\mathcal{F}' \subseteq \mathcal{F}$ of size t. The ground set of $\mathcal{F}'$ also has K as forbidden homological minor. Moreover, $\mathcal{F}'$ has $(\dim K)$-level topological complexity at most $\phi_{\mathcal{F}}^{(\dim K)}(t) \leq S_K(t)$. It follows that $\mathrm{r}_{\mathcal{F}'} \leq \Psi^{(K)}_{\mathrm{hc}\to\mathrm{r}}(S_K(t)) \leq \frac{1}{2}\log_2 t$. This holds for every $\mathcal{F}' \subseteq \mathcal{F}$ of size t, so $\mathrm{r}_{\mathcal{F}}(t) \leq \frac{1}{2}\log_2 t$. This inequality holds for every $t \in \mathbb{N}$ so Theorem 5 implies that $\mathcal{F}$ has bounded Radon number. ◀

We can finally prove a fractional Helly theorem for diverging homological shatter functions.

Proof of Corollary 6. Recall that $\Psi_{\mathrm{r}\to\mathrm{fh}}(y)$ denotes the supremum of the fractional Helly number of a set system with Radon number y. Holmsen and Lee [17, Theorem 1.1] proved that $\Psi_{\mathrm{r}\to\mathrm{fh}}(y)$ is finite for every y. Let S_K denote the function from Corollary 12. Now consider a set system $\mathcal{F}$ whose ground set has K as forbidden homological minor and satisfies $\phi_{\mathcal{F}}^{(\dim K)} \leq S_K$. Corollary 12 ensures that $\mathrm{r}_{\mathcal{F}}$, the Radon number of $\mathcal{F}$, is finite. It follows that the fractional Helly number of $\mathcal{F}$ is at most $\Psi_{\mathrm{r}\to\mathrm{fh}}(\mathrm{r}_{\mathcal{F}})$, and is therefore finite. From there, [13, Theorem 1.2] ensures that this fractional Helly number is at most $\mu(K) + 1$. ◀

3.3 Other graded parameters and relations

With the intersection hypergraph of $\mathcal{F}$ in mind, we say that a set $\mathcal{G}$ in a set system $\mathcal{F}$ is a *clique* if the intersection of all members of $\mathcal{G}$ is non-empty. The colorful Helly theorem suggests the following parameter:

- The *colorful Helly number* $\mathrm{ch}_{\mathcal{F}}$ of $\mathcal{F}$ is the smallest number of colors m such that for every coloring of a subfamily $\mathcal{F}' \subset \mathcal{F}$ with m colors, if every subfamily that contains exactly one element of each color is a clique, then at least one color class is a clique.

We say that a set $\mathcal{G}$ in a set system $\mathcal{F}$ is a *c-wise clique* if every c-element subset of $\mathcal{G}$ is a clique. Clearly a clique is a c-wise clique, and when $c \geq \mathrm{h}_{\mathcal{F}}$ the converse is true. To analyze set systems with large, infinite or unknown Helly numbers, it is useful to consider variants of the colorful and fractional Helly numbers where cliques are replaced by c-wise cliques:

- The *cth colorful Helly number* $\mathrm{ch}_{\mathcal{F}}^{(c)}$ of $\mathcal{F}$ is the smallest number of colors $m \geq c$ such that for every coloring of a subfamily $\mathcal{F}' \subset \mathcal{F}$ with m colors, if every subfamily of $\mathcal{F}'$ that contains exactly one element of each color forms a c-wise clique, then at least one color class is a c-wise clique.
- The *cth fractional Helly number* $\mathrm{fh}_{\mathcal{F}}^{(c)}$ of $\mathcal{F}$ is the smallest integer s such that there exists a function $\beta_{\mathcal{F}} : (0,1) \to (0,1)$ with the following property: For every finite subfamily $\mathcal{F}' \subset \mathcal{F}$, whenever a fraction α of the s-tuples of $\mathcal{F}'$ forms a c-wise clique, a subset $\mathcal{G}$ of $\mathcal{F}'$ of size $\beta_{\mathcal{F}}(\alpha)|\mathcal{F}'|$ forms a c-wise clique.

Obviously for every set system $\mathcal{F}$, if $c \geq \mathrm{h}_{\mathcal{F}}$, then $\mathrm{fh}_{\mathcal{F}}^{(c)} = \mathrm{fh}_{\mathcal{F}}$ and $\mathrm{ch}_{\mathcal{F}}^{(c)} = \mathrm{ch}_{\mathcal{F}}$. Holmsen [16, Theorem 1.2] proved that $\mathrm{fh}_{\mathcal{F}}^{(c)} \leq \mathrm{ch}_{\mathcal{F}}^{(c)}$ for every set system $\mathcal{F}$ and every $c \in \mathbb{N}$. A close inspection of that proof provides another bridge between the graded and ungraded parameters:

▶ **Lemma 13.** *Let $\ell > c$ be integers. Every set system $\mathcal{F}$ such that $\mathrm{ch}_{\mathcal{F}}^{(c)}(c\ell) \leq \ell$ satisfies $\mathrm{fh}_{\mathcal{F}}^{(c)} \leq \mathrm{ch}_{\mathcal{F}}^{(c)}(c\ell)$.*

Proof. Let $c < \ell$, let $\mathcal{F}$ be a set system and let $\ell' := \mathrm{ch}_{\mathcal{F}}^{(c)}(c\ell)$. By definition of $\mathrm{ch}_{\mathcal{F}}^{(c)}(\cdot)$, the c-uniform hypergraph recording which c-element subsets of $\mathcal{F}$ form cliques cannot contain a certain pattern on $c\ell'$ vertices, namely the *complete ℓ'-tuples of missing edges* [17, §3]. This is the only property needed to ensure that $\mathrm{fh}_{\mathcal{F}}^{(c)} \leq \ell'$ [16, Theorem 1.2]. ◀

Let $\Psi_{\mathrm{r}\to\mathrm{ch}}^{(c)}(x)$ denote the supremum of the cth colorful Helly number of a set system with Radon number x. Holmsen and Lee [17, Theorem 2.2] proved that $\Psi_{\mathrm{r}\to\mathrm{ch}}^{(c)}(x) \leq \max\left(\Xi(x), c\right)$ for every $c \geq x - 1$, where $\Xi : \mathbb{N} \to \mathbb{N}$ is the function defined by $\Xi(r) := r^{r^{\lceil \log_2 r \rceil} + r\lceil \log_2 r \rceil}$. Applying this inequality to subsets of size t we get:

$$\mathrm{ch}_{\mathcal{F}}^{(c)}(t) \leq \max\left(\Xi\left(\mathrm{r}_{\mathcal{F}}(t)\right), c\right) \qquad \text{for every } \mathcal{F}, c, t \text{ such that } c \geq \mathrm{h}_{\mathcal{F}}(t). \tag{5}$$

Proof of Theorem 7. Let $\Psi : \mathbb{N} \to \mathbb{N}$ and $t_0 \in \mathbb{N}$ such that $\Psi(t) < t + 1$ for every $t \geq t_0$. Also suppose that there exists an integer $t_1 \geq {t_0}^2$ such that $\Xi\left(\Psi(t_1)\right) < \frac{t_1}{t_0}$. We now consider a set system $\mathcal{F}$ such that $\mathrm{r}_{\mathcal{F}}(t) \leq \Psi(t)$ for every $t \in \mathbb{N}$ and argue that $\mathrm{fh}_{\mathcal{F}}$, the fractional Helly number of $\mathcal{F}$, is bounded.

By Levi's inequality (4) we have $\mathrm{h}_{\mathcal{F}}(t) \leq \mathrm{r}_{\mathcal{F}}(t) - 1 \leq \Psi(t) - 1$. It follows that $\mathrm{h}_{\mathcal{F}}(t) < t$ for every $t \geq t_0$, so by Lemma 10 we have $\mathrm{h}_{\mathcal{F}} \leq t_0$. Hence, the graded version (5) of the Holmsen-Lee inequality applies with $c = t_0$ and every t, that is $\mathrm{ch}_{\mathcal{F}}^{(t_0)}(t) \leq \max\left(\Xi\left(\Psi(t)\right), t_0\right)$.

Let us apply Lemma 13 with $c = t_0$ and $\ell = \frac{t_1}{t_0}$. Observe that $\ell > c$ holds because ${t_1} > {t_0}^2$, and that $\mathrm{ch}_{\mathcal{F}}^{(c)}(c\ell) \leq \ell$ follows from the assumption that $\Xi\left(\Psi(t_1)\right) \leq \frac{t_1}{t_0}$, as

$$\mathrm{ch}_{\mathcal{F}}^{(c)}(c\ell) = \mathrm{ch}_{\mathcal{F}}^{(t_0)}(t_1) \leq \max\left(\Xi\left(\Psi(t_1)\right), t_0\right) \leq \frac{t_1}{t_0} = \ell.$$

Hence, $\mathrm{fh}_{\mathcal{F}}^{(t_0)} \leq \mathrm{ch}_{\mathcal{F}}^{(t_0)}(t_1)$. Since $\mathrm{h}_{\mathcal{F}} \leq t_0$ we have $\mathrm{fh}_{\mathcal{F}}^{(t_0)} = \mathrm{fh}_{\mathcal{F}}$ and the statement follows. ◀

References

1 Noga Alon, Gil Kalai, Jirí Matoušek, and Roy Meshulam. Transversal numbers for hypergraphs arising in geometry. *Advances in Applied Mathematics*, 29:79–101, 2002. doi:10.1016/S0196-8858(02)00003-9.

2 Sergey Avvakumov, Marguerite Bin, and Xavier Goaoc. Intersection patterns of set systems on manifolds with slowly growing homological shatter functions, 2026. arXiv:2601.02920.

3 Ervin G. Bajmóczy and Imre Bárány. On a common generalization of borsuk's and radon's theorem. *Acta Mathematica Hungarica*, 34(3-4):347–350, 1979.

4 Imre Bárány. *Combinatorial convexity*, volume 77. American Mathematical Soc., 2021.

5 Imre Bárány and Pablo Soberón. Tverberg's theorem is 50 years old: a survey. *Bulletin of the American Mathematical Society*, 55(4):459–492, 2018.

6 Boris Bukh. Radon partitions in convexity spaces, 2010. arXiv:1009.2384.

7 Sourav Chakraborty, Rameshwar Pratap, Sasanka Roy, and Shubhangi Saraf. Helly-type theorems in property testing. *International Journal of Computational Geometry & Applications*, 28(04):365–379, 2018. doi:10.1142/S0218195918500115.

8 Jesús De Loera, Xavier Goaoc, Frédéric Meunier, and Nabil Mustafa. The discrete yet ubiquitous theorems of carathéodory, helly, sperner, tucker, and tverberg. *Bulletin of the American Mathematical Society*, 56(3):415–511, 2019.

9 Éric Colin de Verdière, Vojtěch Kaluža, Pavel Paták, Zuzana Patáková, and Martin Tancer. A direct proof of the strong hanani-tutte theorem on the projective plane. *Journal of Graph Algorithms and Applications*, 21(5):939–981, 2017. doi:10.7155/JGAA.00445.

10 Jürgen Eckhoff. The partition conjecture. *Discrete Mathematics*, 221(1):61–78, 2000. doi:10.1016/S0012-365X(99)00386-6.

11 Rogers Epstein and Sandeep Silwal. Property Testing of LP-Type Problems. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 98:1–98:18, 2020. doi:10.4230/LIPIcs.ICALP.2020.98.

12 Radoslav Fulek and Jan Kynčl. Counterexample to an extension of the hanani-tutte theorem on the surface of genus 4. *Combinatorica*, 39(6):1267–1279, 2019. doi:10.1007/S00493-019-3905-7.

13 Xavier Goaoc, Andreas F. Holmsen, and Zuzana Patáková. Intersection patterns in spaces with a forbidden homological minor, 2024. arXiv:2103.09286.

14 Xavier Goaoc, Pavel Paták, Zuzana Patáková, Martin Tancer, and Uli Wagner. Bounding Helly numbers via Betti numbers. In *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 407–447. Springer, 2017.

15 Eduard Helly. Über Systeme von abgeschlossenen Mengen mit gemeinschaftlichen Punkten. *Monatshefte für Mathematik und Physik*, 37:281–302, 1930.

16 Andreas F. Holmsen. Large cliques in hypergraphs with forbidden substructures. *Combinatorica*, 40(4):527–537, August 2020. doi:10.1007/S00493-019-4169-Y.

17 Andreas F. Holmsen and Donggyu Lee. Radon numbers and the fractional Helly theorem. *Israel Journal of Mathematics*, 241(1):433–447, March 2021.

18 Gil Kalai. Intersection patterns of convex sets. *Israel Journal of Mathematics*, 48(2):161–174, 1984.

19 Gil Kalai. Combinatorial expectations from commutative algebra. In I. Peeva anv V. Welker, editor, *Combinatorial Commutative Algebra*, volume 1(3), pages 1729–1734. Oberwolfach Reports, 2004.

20 Meir Katchalski and Andy Liu. A problem of geometry in $\mathbb{R}^n$. *Proceedings of the American Mathematical Society*, 75(2):284–288, 1979.

21 Friedrich W. Levi. On Helly's theorem and the axioms of convexity. *Journal of the Indian Mathematical Society*, 15:65–76, 1951.

22 Jiři Matoušek. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2013.

23 Jirí Matoušek. A Helly-type theorem for unions of convex sets. *Discrete & Computational Geometry*, 18:1–12, 1997. doi:10.1007/PL00009305.

24 Jirí Matoušek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996. doi:10.1007/BF01940877.

25 Jirí Matoušek. Bounded VC-dimension implies a fractional Helly theorem. *Discrete & Computational Geometry*, 31:251–255, 2004. doi:10.1007/S00454-003-2859-Z.

26 Pavel Paták. A sharper ramsey theorem for constrained drawings. *Journal of Graph Theory*, 109(4):401–411, 2025.

27 Pavel Paták and Martin Tancer. Embeddings of k-complexes into 2 k-manifolds. *Discrete & Computational Geometry*, 71(3):960–991, 2024. doi:10.1007/S00454-023-00595-W.

28 Zuzana Patáková. Bounding Radon number via Betti numbers. *International Mathematics Research Notices*, 2024(11):9482–9500, 2024.

29 Michael J Pelsmajer, Marcus Schaefer, and Despina Stasi. Strong hanani–tutte on the projective plane. *SIAM Journal on Discrete Mathematics*, 23(3):1317–1323, 2009. doi:10.1137/08072485X.

30 Victor V. Prasolov. *Elements of homology theory*, volume 81. American Mathematical Society, 2007.

31 Gerhard Ringel. *Map color theorem*, volume 209. Springer Science & Business Media, 2012.

32 Colin Rourke and Brian Sanderson. *Introduction to Piecewise-Linear Topology*. Springer Study Edition. Springer Berlin, Heidelberg, 1972.

33 Arkadiy Skopenkov. Embeddings of k-complexes in $2k$-manifolds and minimum rank of partial symmetric matrices, 2024. arXiv:2112.06636.

34 Martin Tancer. Intersection patterns of convex sets via simplicial complexes: a survey. In *Thirty essays on geometric graph theory*, pages 521–540. Springer, 2012.

35 Uli Wagner. Minors, embeddability, and extremal problems for hypergraphs. In *Thirty Essays on Geometric Graph Theory*, pages 569–607. Springer, 2012.

Fast Free Resolutions of Bifiltered Chain Complexes

Ulrich Bauer
Department of Mathematics and MDSI and MCML, Technical University of Munich, Germany

Tamal K. Dey
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Michael Kerber
Institute of Geometry, Graz University of Technology, Austria

Florian Russold
Institute of Geometry, Graz University of Technology, Austria

Matthias Söls
Department of Mathematics, Technical University of Munich, Germany
Institute of Geometry, Graz University of Technology, Austria

Abstract

In a k-critical bifiltration, every simplex enters along a staircase with at most k steps. Examples with $k > 1$ include degree-Rips bifiltrations and models of the multicover bifiltration. We consider the problem of converting a k-critical bifiltration into a 1-critical (i.e. free) chain complex with equivalent homology. This is known as computing a free resolution of the underlying chain complex and is a first step toward post-processing such bifiltrations.

We present two algorithms. The first one computes free resolutions corresponding to path graphs and assembles them to a chain complex by computing additional maps. The simple combinatorial structure of path graphs leads to good performance in practice, as demonstrated by extensive experiments. However, its worst-case bound is quadratic in the input size because long paths might yield dense boundary matrices in the output. Our second algorithm replaces the simplex-wise path graphs with ones that maintain short paths which leads to almost linear runtime and output size.

We demonstrate that pre-computing a free resolution speeds up the task of computing a minimal presentation of the homology of a k-critical bifiltration in a fixed dimension. Furthermore, our findings show that a chain complex that is minimal in terms of generators can be asymptotically larger than the non-minimal output complex of our second algorithm in terms of description size.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases Topological Data Analysis, Multi-Parameter Persistence, Multi-Critical Bifiltrations

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.10

Related Version *Full Version*: `https://arxiv.org/abs/2512.08652` [3]

Supplementary Material *Software*: `https://doi.org/10.3217/y06aq-p6a53`

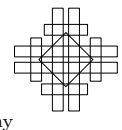
Funding *Tamal K. Dey*: NSF grants DMS-2301360 and CCF-2437030.
Michael Kerber: Austrian Science Fund (FWF) grant 10.55776/P33765.
Florian Russold: Austrian Science Fund (FWF) grants 10.55776/P33765 and 10.55776/W1230.
Matthias Söls: Austrian Science Fund (FWF) grants 10.55776/P33765 and 10.55776/W1230.

1 Introduction

Motivation and problem statement. Multi-parameter persistence is a branch of topological data analysis where a data set (e.g., a point cloud) is filtered with respect to two or more parameters and the topological evolution of the data when changing the parameters is



© Ulrich Bauer, Tamal K. Dey, Michael Kerber, Florian Russold, and Matthias Söls;
licensed under Creative Commons License CC-BY 4.0





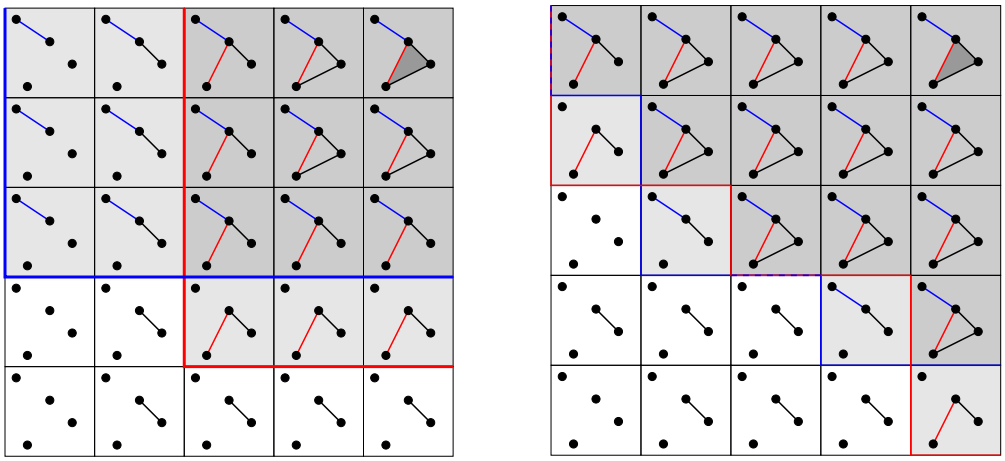


Figure 1 The bifiltration on the left is 1-critical while the bifiltration on the right is 3-critical. The support of every simplex is an upset, as visualized for two edges (red and blue, respectively).

analyzed. In this context, the first step of a computational pipeline for two parameters typically consists of the computation of a *bifiltration* of simplicial complexes, that is, a family of simplicial complexes indexed by $\mathbb{N}^2$ that grows when increasing the parameters. We refer to the parameter set $\mathbb{N}^2$ as the *grades* of the bifiltration. A bifiltration can be equivalently described by determining the *support* of each simplex, that is, the set of grades at which the simplex is part of the complex. A bifiltration is called *free* or 1-*critical* if the support of every simplex is a *principal upset*, that is, the upward closure of a single element in the parameter space. More generally, a bifiltration is k-*critical* if the support of a simplex is the upward closure of at most k elements; see Figure 1 for a visualization.

A bifiltration gives rise to a bifiltered chain complex $C_\bullet$

$$0 \longleftarrow C_0 \xleftarrow{\partial_1} C_1 \longleftarrow \cdots \longleftarrow C_{d-1} \xleftarrow{\partial_d} C_d$$

with the k-simplices at a given grade forming the basis of the k-chains at that grade. Free and k-critical filtered chain complexes are defined analogously.

Bifiltered chain complexes arise naturally in various constructions for topological data analysis, and several constructions produce bifiltrations that are not free. The most prominent example is given by the degree-Rips bifiltration [19, 23, 24, 7]. Other examples are the approximate multi-cover bifiltration [1] and Delaunay core bifiltration [6]. On the other hand, free bifiltrations are most suitable for algorithmic and computational purposes. There are fast algorithms for minimizing a bifiltered chain complex without changing its homology [14, 15, 22] as well as for computing the homology of a chain complex in terms of a minimal presentation [20, 18, 15, 4], but both approaches require free chain complexes as input.

In homological algebra, a standard technique is to replace a general chain complex by a free one, connected to the original chain complex by a chain map that induces an isomorphism in homology (a *quasi-isomorphism*). The free chain complex together with the chain map is called a *free resolution* of the original complex. The problem studied in this paper is how to efficiently compute a free resolution of a non-free bifiltered chain complex.

At the homology level, one way to address the case of a non-free bifiltration was proposed by Chacholski, Scolamiero and Vaccarino [8]: given a segment $C : C_{m+1} \to C_m \to C_{m-1}$ of a chain complex, the authors describe an algorithm to compute a free chain complex $F : F_{m+1} \to F_m \to F_{m-1}$ such that $H_m(C)$ and $H_m(F)$ are isomorphic, that is, a *free*

implicit representation. This algorithm suffices if one is interested in a presentation of $H_m(C)$ for further processing. Methods to compute a projective implicit representation from families of simplicial complexes and general simplicial maps have been developed in [11, 12], where, as in our approach, graph theoretic methods are used to speed up computations.

There are good reasons to work on the level of entire chain complexes instead. First of all, there are potential computational advantages, especially if one is interested in multiple homology dimensions (see Section 6). Moreover, the chain complex structure can encode subtle information on the data that is lost at the homology level: for example, two chain complexes may have isomorphic homology in every dimension without being quasi-isomorphic.

Computing free resolutions of general chain complexes is a standard task in computational algebra, available in the computer algebra software Macaulay2 for a much wider range of chain complexes. In the context of applied topology, we are mostly interested in very large bifiltered chain complexes with millions of generators but a simple combinatorial structure (i.e., simplicial boundary maps). The goal of this paper is to develop specialized and highly optimized algorithms for this type of input data, which the general purpose algorithms implemented in existing computer algebra systems are not tailored for.

Contributions. Our main contribution is to propose two algorithms to compute a free resolution of a bifiltered chain complex. Both algorithms rely on the same simple idea of expanding any k-critical simplex in dimension p into a sequence of k free copies, with consecutive copies related by $(p+1)$-dimensional elements at the join of their grades. In algebraic terms, this corresponds to a free resolution of the upset module associated to the simplex. In order to construct a valid total complex, the algorithms introduce further maps to establish the chain complex property while maintaining quasi-isomorphism to the original filtered complex. Finding these maps is computationally inexpensive and takes place at a purely combinatorial level.

The algorithms differ mainly in the choice of free resolution of the upset modules. In the first algorithm, the *path algorithm*, the free resolution corresponds to the chain complex of a filtered path graph (as a simple special case of a cellular resolution [5, 21]). We give an example of a family of simplicial bifiltrations so that the resulting free chain complex has a dense boundary matrix (for any choice of basis). This example shows tightness of the worst-case runtime $O(n^2)$, where n is the number of input generators (description size).

The second algorithm, the *log-path algorithm*, extends the path algorithm by adding additional relations to the free resolution of a simplex, such that any pair of copies of a simplex is connected via a sequence of relations of logarithmic length. This ensures sparsity of the boundary matrices in the output chain complex, but also requires adding further higher relations (*syzygies*). Again, further maps are required to establish the chain complex property of the resulting total complex, allowing the algorithm to maintain sparsity and obtain a resolution with worst-case run time in $O(n \log^2 n)$.

Our findings lead to an interesting observation: as shown by our worst case example, a minimal resolution may require dense boundary matrices, while a non-minimal resolution may actually admit a sparse matrix representation, with asymptotically fewer non-zero entries ($O(n \log^2 n)$ instead of $O(n^2)$, where n is the description size of the complex). This observation suggests that minimizing chain complexes does not necessarily speed-up subsequent algorithmic tasks, at least in certain worst-case examples.

We provide implementations of the path and log-path algorithms, in addition to the Chacholski–Scolamiero–Vaccarino algorithm for computing free implicit representations of homology. Systematic tests on various k-critical bifiltrations show that the overhead of

the log-path algorithm over the path algorithm does not exceed a factor of 3 in run time and a factor of 2 in the number of non-zero entries for our examples, while showing the expected improvement on the mentioned worst-case examples. Subsequently minimizing the free chain complex yields a further significant reduction of the size. Furthermore, we consider the task of computing minimal presentations of homology in all degrees, comparing the approach of first computing a "global" free resolution with the approach of computing free implicit representations. Our results show a clear computational advantage for the global approach. Remarkably, for some instances, computing *all* minimal presentations using a free resolution is faster than computing a *single* minimal presentation using the Chacholski–Scolamiero–Vaccarino algorithm.

2 Bifiltered chain complexes

Bifiltrations. A *simplicial bifiltration* $\mathcal{K}$ is an abstract simplicial complex together with a collection of subcomplexes $(\mathcal{K}_s)_{s\in\mathbb{N}^2}$ such that $\mathcal{K}_s \subseteq \mathcal{K}_t$ whenever $s \leq t$ (which means that $s_1 \leq t_1$ and $s_2 \leq t_2$). As shown in Figure 1, each simplex σ enters $\mathcal{K}$ along a staircase bounding the *support* of σ, denoted $\operatorname{supp}(\sigma)$, which is an *upset* (an upward closed subset of $\mathbb{N}^2$). There is a unique minimal set of grades $\mathcal{G}(\sigma) := \{x_1, \ldots, x_m\} \subseteq \mathbb{N}^2$ such that $\operatorname{supp}(\sigma) = \{s \in \mathbb{N}^2 \mid \exists x_i \leq s\}$. If $m \leq k$ for each $\sigma \in \mathcal{K}$, then the bifiltration is *k-critical*.

Bipersistence modules. Simplicial bifiltrations give rise to bifiltered chain complexes. We first describe the elementary building blocks. Each p-simplex σ has a support $\operatorname{supp}(\sigma)$ with minimal generating set $\mathcal{G}(\sigma) = \{x_1, \ldots, x_m\}$, determining an *upset module* $U_{\{x_1,\ldots,x_m\}}$ (which we also denote by U_σ for brevity) given by

$$(U_\sigma)_s := \begin{cases} \Bbbk, & \text{if } \exists\, x_i \leq s, \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad (U_\sigma)_{s,t} = \begin{cases} \mathrm{id}_\Bbbk, & \text{if } \exists\, x_i \leq s, \\ 0 & \text{otherwise} \end{cases}$$

as illustrated in Figure 2. This is a special case of a *bipersistence module* M, which is a family $(M_s)_{s\in\mathbb{N}^2}$ of vector spaces over a field $\Bbbk$ together with a family of homomorphisms $(M_{s,t} \colon M_s \to M_t)_{s\leq t\in\mathbb{N}^2}$. Although our results extend to arbitrary fields straightforwardly, we will stick to the case $\Bbbk = \mathbb{Z}_2$ in order to simplify the exposition. A *morphism of bipersistence modules* $\varphi : M \to N$ is a natural transformation, that is, a family of linear maps $(f_s)_{s\in\mathbb{N}^2}$ that commute with the structure maps of M and N: $f_t \circ M_{s,t} = N_{s,t} \circ f_s$ for $s \leq t$.

A direct sum of upset modules, each generated by a single grade, is called *free*. A *basis* of a free bipersistence module F is a set of elements $\{b^{x_1}, \ldots, b^{x_n}\}$ where each $b^{x_i} \in F_{x_i}$, such that every $v \in F_y$, for any grade y, can be written in unique way as a linear combination of images of the basis elements under the structure maps (which are inclusions into the vector space $\bigcup_z F_z$). Note that the grades appearing in this linear combination with a non-zero coefficient must be less or equal to y. A morphism f between free modules is then determined by the images of the basis elements for the domain, and can be encoded by a matrix $[f]$.

Chain complexes. The upset modules associated to the simplices assemble to

$$0 \longleftarrow C_0 \xleftarrow{\partial_1} C_1 \xleftarrow{\partial_2} \cdots \xleftarrow{\partial_{i-1}} C_{i-1} \xleftarrow{\partial_i} C_i \xleftarrow{\partial_{i+1}} C_{i+1} \longleftarrow \cdots,$$

which is a *bifiltered chain complex* with $C_i := \bigoplus_{\sigma\in\mathcal{K}^{(i)}} U_\sigma$ and *boundary maps* $\partial_i : C_i \to C_{i-1}$ inherited from the simplicial complex $\mathcal{K}$, satisfying $\partial_i \circ \partial_{i+1} = 0$. The notion of a k-critical bifiltered chain complex is defined analogously to the setting of a simplicial bifiltration.

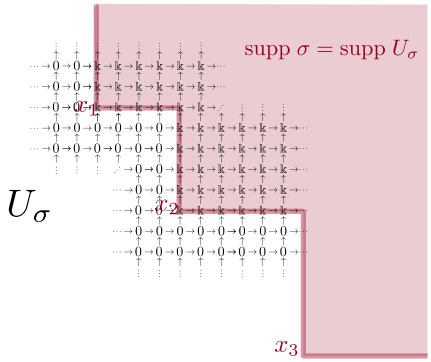


Figure 2 The upset module U_σ induced by the simplex σ.

Even though a bifiltered chain complex has a simple combinatorial structure, a *free chain complex* (a chain complex $F_\bullet$ with each F_i free) is preferred in computational (and algebraic) settings. This motivates the goal of finding a *free resolution*, that is, a free chain complex together with a chain map $F_\bullet \to C_\bullet$ that induces an isomorphism $H_i(F_\bullet) \cong H_i(C_\bullet)$ on homology for each $i \geq 0$. Such a chain map is called a *quasi-isomorphism*.

Data representation. The list of all simplices of a bifiltration $\mathcal{K}$, each with a lexicographically ordered list of the minimal generating set of its support, and a list of its facets gives a full combinatorial description of $\mathcal{K}$ and will serve as the *input* to our algorithms. This data representation generalizes to regular cell complexes, and in our case to bifiltered chain complexes where $\Bbbk = \mathbb{Z}_2$. For the simplicial case, the number of facets is constant for each simplex dimension, ensuring sparsity. Therefore the description size of the input is given, up to a constant factor, by the total number of generators in this list. Our *output* is a free resolution $F_\bullet$ of our input chain complex $C_\bullet$, again represented in the same data format.

3 Free implicit representations of homology

FI-reps. To compute the homology $H_m(C_\bullet)$ of a k-critical chain complex in a chosen dimension m, Chacholski, Scolamiero, and Vaccarino [8] provide a procedure to construct a free chain complex segment $X \xleftarrow{f} Y \xleftarrow{g} Z$ from the input segment

$$C_{m-1} \xleftarrow{\partial_m} C_m \xleftarrow{\partial_{m+1}} C_{m+1}. \tag{1}$$

such that $H_m(C_\bullet) \cong \ker f / \operatorname{im} g$. The pair of graded matrices $([f], [g])$ is called a *free implicit representation (FI-rep)* of $H_m(C_\bullet)$ [20], and it serves as input for the computation of minimal presentations of $H_m(C_\bullet)$ [20, 18].

The Chacholski–Scolamiero–Vaccarino algorithm. We consider an input complex $C_\bullet$ induced by a simplicial bifiltration $\mathcal{K}$. The first step is to extend each U_σ in C_{m-1}, induced by an $(m-1)$-simplex σ, to a free upset module $U_{\{0\}}$. The resulting free module $D_{m-1} := \bigoplus_{\sigma \in \mathcal{K}^{(m-1)}} U_{\{0\}}$, whose basis we denote by $\{b^0_\sigma\}_{\sigma \in \mathcal{K}^{(m-1)}}$, contains C_{m-1} as a submodule, since the support of U_σ is contained in the support of $U_{\{0\}}$. Thus, postcomposing ∂_m with this submodule inclusion does not affect the kernel: we have $\ker \partial_m = \ker \iota \circ \partial_m$.

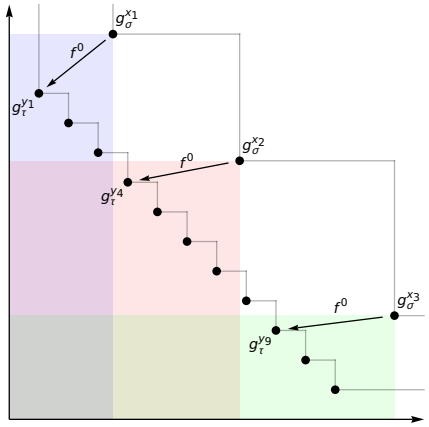


Figure 3 f^0_{m+1} sends each generator g^x_σ to a generator g^y_τ of the facet τ of σ whose grade y is in the downset of x (say, with the smallest first coordinate).

As a second step, we *cover* each U_σ in C_{m+1}, induced by an $(m+1)$-simplex σ by a free bipersistence module. This means that we replace each upset module U_σ by the free module $G_\sigma \coloneqq \bigoplus_{x_i \in \mathcal{G}(\sigma)} U_{\{x_i\}}$, connected to the upset module by a surjection $G_\sigma \twoheadrightarrow U_\sigma$ since the support of G_σ equals the support of U_σ. We call the basis elements $g_\sigma^{x_i}$ the *generators* of U_σ. The module G_{m+1} arises from C_{m+1} by covering U_σ by G_σ for each $(m+1)$-simplex via a canonical surjection $G_{m+1} \xrightarrow{\alpha_{m+1}} C_{m+1}$. Hence, precomposing ∂_{m+1} with α_{m+1} does not affect the image. The input (1) can now be replaced by (2) which has isomorphic homology.

$$D_{m-1} \xleftarrow{\iota \circ \partial_m} C_m \xleftarrow{\partial_{m+1} \circ \alpha_{m+1}} G_{m+1}, \tag{2}$$

As the third and final step it remains to replace C_m. We first construct G_m analogously to G_{m+1}, by substituting each U_σ in C_m by G_σ. Precomposing $\iota \circ \partial_m$ with the surjection $G_m \xrightarrow{\alpha_m} C_m$ yields a map $\gamma \coloneqq \iota \circ \partial_m \circ \alpha_m \colon G_m \to D_{m-1}$, as depicted in (3). For $\sigma \in \mathcal{K}^{(m)}$ with $\partial\sigma = \tau_0 + \cdots + \tau_m$, the resulting map γ sends a generator g^x_σ in G_m to $b^0_{\tau_0} + \cdots + b^0_{\tau_m}$ in D_{m-1}.

$$\begin{array}{ccccccc} & & & & R_m & & \\ & & & & \downarrow{\scriptstyle p^1_m} & & \\ & & & & G_m & \xleftarrow{f^0_{m+1}} & G_{m+1} \\ & & & \overset{\gamma}{\swarrow} & \downarrow{\scriptstyle \alpha_m} & & \downarrow{\scriptstyle \alpha_{m+1}} \\ D_{m-1} & \xleftarrow{\iota} & C_{m-1} & \xleftarrow{\partial_m} & C_m & \xleftarrow{\partial_{m+1}} & C_{m+1} \end{array} \tag{3}$$

Replacing C_m by G_m makes it necessary to also replace the map $\partial_{m+1} \circ \alpha_{m+1} \colon G_{m+1} \to C_m$ by a map $f^0_{m+1} \colon G_{m+1} \to G_m$, representing ∂_{m+1} on the generators and also making the diagram commute. This map f^0_{m+1} sends each generator g^x_σ of an $(m+1)$-simplex σ with $\partial\sigma = \tau_0 + \cdots + \tau_m$ to $f^0_{m+1}(g^x_\sigma) = g^{y_0}_{\tau_0} + \cdots + g^{y_m}_{\tau_m}$, where each $g^{y_i}_{\tau_i}$ is a generator of τ_i chosen such that its grade satisfies $y_i \leq x$; such a y_i always exists because the faces τ_i of a simplex σ are present in the bifiltration whenever σ itself is present. The map f^0_{m+1} thus makes the square in (3) commute and is therefore called a *lift* of ∂_{m+1}. See Figure 3 for an illustration.

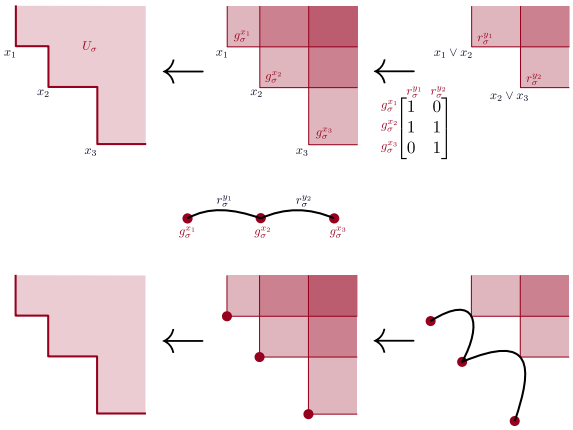


Figure 4 The top row illustrates the free resolution of U_σ as in Diagram 4 and the middle row shows its path resolution. We often visualize both at once as in the bottom row.

Note that the kernel of γ may not be isomorphic to the kernel of ∂_m; the surjection α_m maps each generator g_σ^x of an m-simplex in G_m to the same potential cycle in C_m, thus increasing the dimension of the kernel. This can be resolved by relating these generators appropriately. Assume that the generators $g_\sigma^{x_1}, \dots, g_\sigma^{x_k}$ of U_σ are ordered w.r.t. the first coordinate of their grades. To represent U_σ correctly, two consecutive generators $g_\sigma^{x_i}$ and $g_\sigma^{x_{i+1}}$ have to be identified at the join $y_i = x_i \vee x_{i+1}$ of their grades by a *relation* $r_\sigma^{y_i}$. With the free bipersistence module $R_\sigma \coloneqq \bigoplus_{y_i} U_{\{y_i\}}$ with basis $\{r_\sigma^{y_i}\}_i$, we can present U_σ by generators and relations via

$$0 \longleftarrow U_\sigma \xleftarrow{\alpha_\sigma} G_\sigma \xleftarrow{p_\sigma^1} R_\sigma, \tag{4}$$

where $p_\sigma^1(r_\sigma^{y_i}) = g_\sigma^{x_i} + g_\sigma^{x_{i+1}}$. Since p_σ^1 is injective, α_σ is surjective, and $\ker \alpha_\sigma = \operatorname{im} p_\sigma^1$, the sequence in (4) is a short exact sequence. This means that $(G_\sigma \xleftarrow{p_\sigma^1} R_\sigma, \alpha_\sigma)$ already determines a *free resolution* of U_σ, as defined in Section 2, where U_σ is considered as a chain complex concentrated in degree 0.

The free resolution in Diagram 4 has a simple combinatorial structure, given by a path graph $\mathcal{P}_\sigma$ with k vertices corresponding to the generators $g_\sigma^{x_i}$ and $k-1$ edges $\{g_\sigma^{x_i}, g_\sigma^{x_{i+1}}\}$ corresponding to the relations $r_\sigma^{y_i}$. Note that vertices and edges implicitly carry the grades of the generators and relations. The morphism p_σ^1 is represented by the (graded) incidence matrix of the graph. We call this free resolution the *path resolution* of U_σ, see Figure 4. Note that a path resolution is a special case of a *cellular resolution* [5, 21].

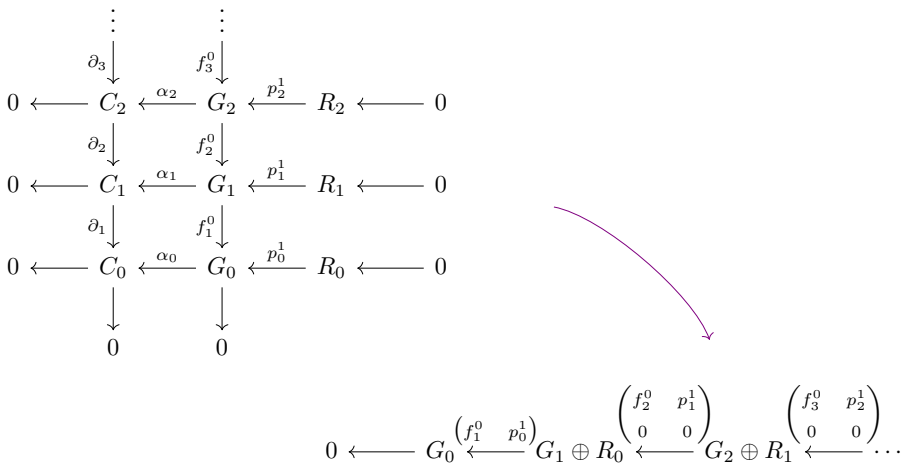
We now use the free resolution to replace the map $\partial_{m+1} \circ \alpha_{m+1}$ in (2), assembling the following output complex from (3)

$$D_{m-1} \xleftarrow{\gamma_m} G_m \xleftarrow{\begin{pmatrix} f_{m+1}^0 & p_m^1 \end{pmatrix}} G_{m+1} \oplus R_m. \tag{5}$$

By construction, the homology of this complex segment in degree m is isomorphic to that of (2), and hence to that of the input complex.

4 Path algorithm

Description. We now turn to the problem of finding a *free resolution*, that is, a chain complex of free modules that is quasi-isomorphic to the input $C_\bullet$, thus preserving homology in *all* dimensions. For this problem, extending the codomain of $\partial_i : C_i \to C_{i-1}$ to a free module generated in degree 0, as done in the Chacholski–Scolamiero–Vaccarino algorithm, is not feasible anymore, since this changes $H_{i-1}(C_\bullet)$. Instead, we now carry out the step of substituting C_i by its free resolution $G_i \xleftarrow{p_i^1} R_i \leftarrow 0 \leftarrow \cdots$ in every dimension i and replacing ∂_{i+1} by the (zeroth) lifts $f_{i+1}^0 \colon G_{i+1} \to G_i$, yielding maps $\begin{pmatrix} f_{i+1}^0 & p_i^1 \end{pmatrix}$ as illustrated in (6).



(6)

However, the sequence of maps, arising from this construction, does not yield a chain complex. Given a relation $r_\sigma^{y_j}$ in R_i, we obtain

$$f_i^0 \circ p_i^1(r_\sigma^{y_j}) = f_i^0(g_\sigma^{x_j}) + f_i^0(g_\sigma^{x_{j+1}}) = g_{\tau_0}^{z_1} + \cdots + g_{\tau_i}^{z_i} + g_{\tau_0}^{z_1'} + \cdots + g_{\tau_i}^{z_i'}.$$

This term (also the term $f_{i-1}^0 \circ f_i^0$) may not vanish, since the generators of σ may be sent to different generators of its boundary simplices τ_j by f_i^0, i.e., $g_{\tau_j}^{z_j} \neq g_{\tau_j}^{z_j'}$. However, it vanishes modulo relations, in the sense that its representative vanishes in $G_{i-1}/\operatorname{im} p_{i-1}^1$, see Lemma 1.

▶ **Lemma 1** (Path Lemma). *For any $g_\tau^{x_j}$, $g_\tau^{x_\ell}$ in G_{i-1}, there is a unique set of distinct relations $r_\tau^{w_j}, \ldots, r_\tau^{w_{\ell-1}}$ in R_{i-1} such that $p_{i-1}^1(r_\tau^{w_j} + \cdots + r_\tau^{w_{\ell-1}}) = g_\tau^{x_j} + g_\tau^{x_\ell}$.*

Proof. This can be seen combinatorially. The free resolution of U_τ corresponds to the path graph $\mathcal{P}_\tau$, in the sense that the restriction of p_{i-1}^1 to a map $R_\tau \to G_\tau$ corresponds to the boundary map of the path graph. The relations $r_\tau^{w_j}, \ldots, r_\tau^{w_{\ell-1}}$ then define the unique path connecting the vertices $g_\tau^{x_j}$ and $g_\tau^{x_\ell}$ in $\mathcal{P}_\tau$. ◀

The map p_i^1 sends the edge $r_\sigma^{y_j}$ of $\mathcal{P}_\sigma$ to the vertices $g_\sigma^{x_j}$ and $g_\sigma^{x_{j+1}}$ which in turn are mapped to $g_{\tau_j}^{z_j}$ and $g_{\tau_j}^{z_j'}$ on $\mathcal{P}_{\tau_j}$ by f_i^0. We can therefore modify the boundary map in (6) by additionally sending $r_\sigma^{y_j}$ to the path $r_{\tau_j}^{w_1} + \cdots + r_{\tau_j}^{w_\ell}$ guaranteed by Lemma 1, see Figure 5. This defines a correction term $f_i^1 : R_i \to R_{i-1}$ such that $p_{i-1}^1 \circ f_i^1 = f_i^0 \circ p_i^1$. The maps f_i^1 are called the *first lifts* of ∂_i.

Even after adding the correction term f_i^1 to (6), the resulting maps may still fail to be boundary maps, since the composition $f_{i-1}^0 \circ f_i^0(g_\sigma^x)$ need not vanish for each generator g_σ^x in G_i, as seen in Figure 6. Again, $f_{i-1}^0 \circ f_i^0(g_\sigma^x)$ is a sum consisting of pairs $g_\tau^y, g_\tau^{y'}$ of possibly

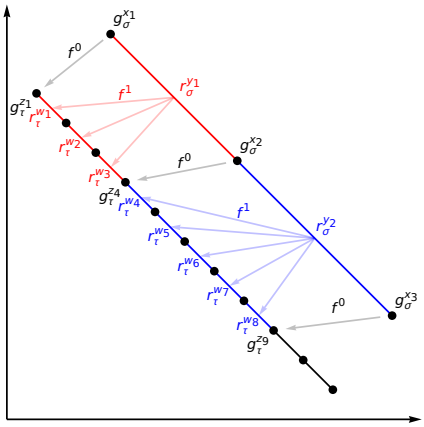
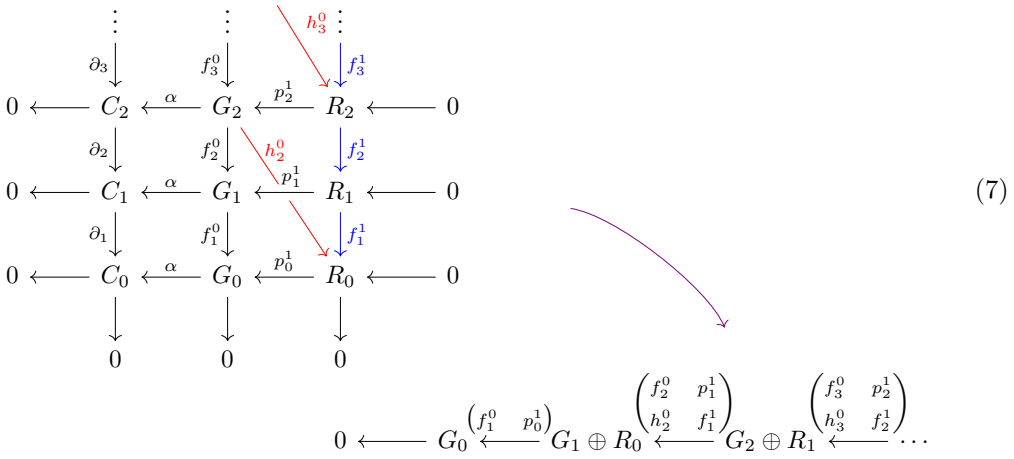


Figure 5 f_i^1 sends each edge r_σ^y to the path connecting the images of its endpoints under f_i^0.

different generators of the same simplex τ, which can be connected by a path of relations as above. A second correction term $h_i^0 \colon G_i \to R_{i-2}$ is defined analogously to the lifts f_i^1. By construction, h_i^0 satisfies $p_{i-2}^1 \circ h_i^0 = f_{i-1}^0 \circ f_i^0$, making the following diagram commute:



(7)

Adding h_i^0 and f_i^1 to the boundary maps in (6) yields the desired free resolution of $C_\bullet$. The next theorem, proved in A.1 of the full version [3], shows correctness of the path algorithm.

▶ **Theorem 2.** *The chain complex on the right side of* (7) *is a free resolution of* $C_\bullet$.

Path algorithm. For the description of the algorithm and its asymptotic time complexity, we assume that the input $C_\bullet$ is induced by a k-critical bifiltration of a simplicial complex with fixed dimension d, given as a list of simplices σ, each with its dimension, grades $\mathcal{G}(\sigma)$ and boundary $\partial\sigma$. The output is described by a list of generators and relations (each carrying a grade) as well as their images under the maps p_i^1, f_i^0, f_i^1 and h_i^0 (which together form the boundary maps of the free resolution). The output is initialized as an empty list.

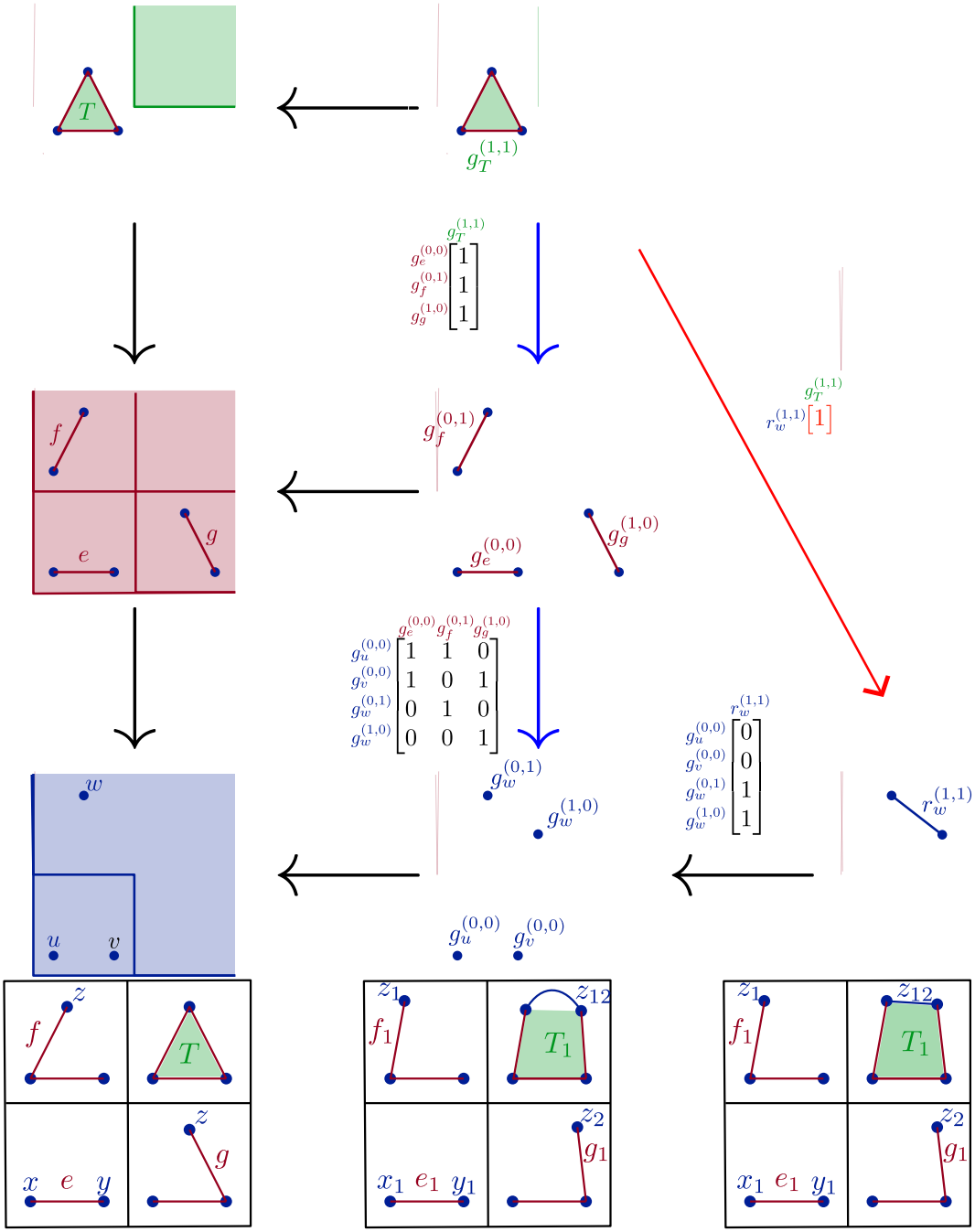


Figure 6 Construction of Diagram 7 for the bifiltration on the bottom left. The boundary defect $f_0^1 \circ f_1^1 \neq 0$, can be repaired by adding the red correction morphism which corresponds to a change of the boundary of T_1 in the bottom-middle cell complex. This yields the bottom-right output.

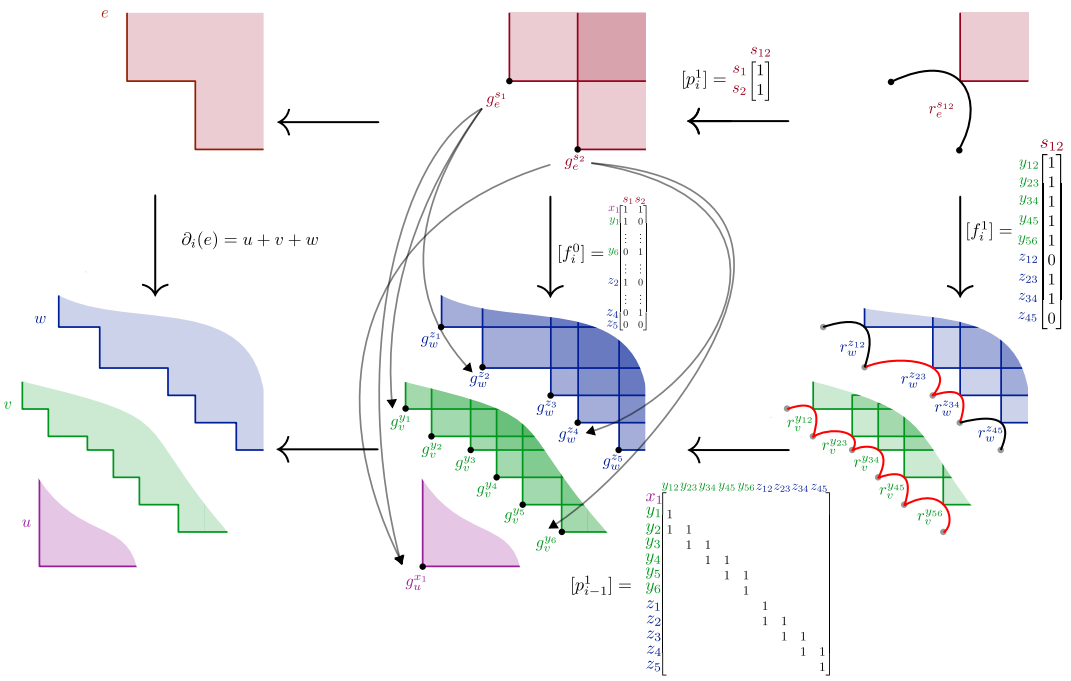


Figure 7 Construction of f_i^1. The relation $r_e^{s_{12}}$ is mapped to the paths $r_w^{z_{23}} + r_w^{z_{34}}$ and $r_v^{y_{12}} + \cdots + r_v^{y_{56}}$ in the path resolutions of w and v, highlighted in red.

1. Compute the bases of $G_i \oplus R_{i-1}$ and the matrices $[p_i^1]$. For each i-simplex σ and $x_j \in \mathcal{G}(\sigma)$, add a generator $g_\sigma^{x_j}$ to G_i. For any two consecutive $x_j, x_{j+1} \in \mathcal{G}(\sigma)$, add a relation $r_\sigma^{y_j}$ with $y_j = x_j \vee x_{j+1}$ to R_i. Moreover, define $p_i^1(r_\sigma^{y_j}) := g_\sigma^{x_j} + g_\sigma^{x_{j+1}}$.
2. Compute the matrices $[f_i^0]$. For each generator g_σ^x in G_i, find generators $g_{\tau_0}^{x_0}, \ldots, g_{\tau_i}^{x_i}$ for $\tau_0 + \cdots + \tau_i = \partial\sigma$ such that $x_j \leq x$ and define $f_i^0(g_\sigma^x) := g_{\tau_0}^{x_0} + \cdots + g_{\tau_i}^{x_i}$.
3. Compute $[f_i^1]$ and $[h_i^0]$. For $[f_i^1]$, perform a matrix multiplication $[f_i^0][p_i^1]$. The resulting columns are indexed by the relations r_σ^y in R_i and consist of one pair of generators $g_\tau^{z_j}, g_\tau^{z_\ell}$ for each facet τ of σ. Such a pair is connected by a path $c_\tau := r_\tau^{w_j} + \cdots + r_\tau^{w_\ell}$ in $\mathcal{P}_\tau$, and we set $f_i^1(r_\sigma^y) := \sum_\tau c_\tau$; see Figure 7. For $[h_i^0]$, proceed analogously to multiply $[f_{i-1}^0][f_i^0]$. The resulting columns are indexed by the generators g_σ^x in G_i and consist of one pair of generators $g_\tau^{z_j}, g_\tau^{z_\ell}$ in G_{i-2} for each codimension 2 face τ of σ. Such a pair is connected by a path $c_\tau := r_\tau^{w_j} + \cdots + r_\tau^{w_\ell}$ in $\mathcal{P}_\tau$, and we set $h_i^0(g_\sigma^x) := \sum_\tau c_\tau$.

Complexity. The running time of the path algorithm is $O(nk)$ for a k-critical bifiltration of description size n; see A.2. in the full version [3]. This runtime is worst-case optimal, since the description size of a computed free resolution can be $\Omega(nk)$ in the worst-case. The reason is that the paths in the computation of f_i^1 and h_i^0 can be of size up to k, and many simplices may require such long paths in the worst case. Figure 8 gives a construction for which the matrix $[h_2^0]$ is of size $\Omega(nk)$; see Figure 15 in [3] for a similar construction for $[f_1^1]$.

A free chain complex is *minimal* if there is no quasi-isomorphic free chain complex with a smaller number of generators. In general, free resolutions computed by the path algorithm are not minimal. However, a slight modification of the construction in Figure 8 yields the following result (see A.3. in the full version [3] for details).

▶ **Proposition 3.** *There is a k-critical simplicial bifiltration of description size n for which the path algorithm computes a free resolution represented by matrices of description size $\Omega(nk)$. This free resolution is minimal, and no other choice of basis yields a smaller description size.*

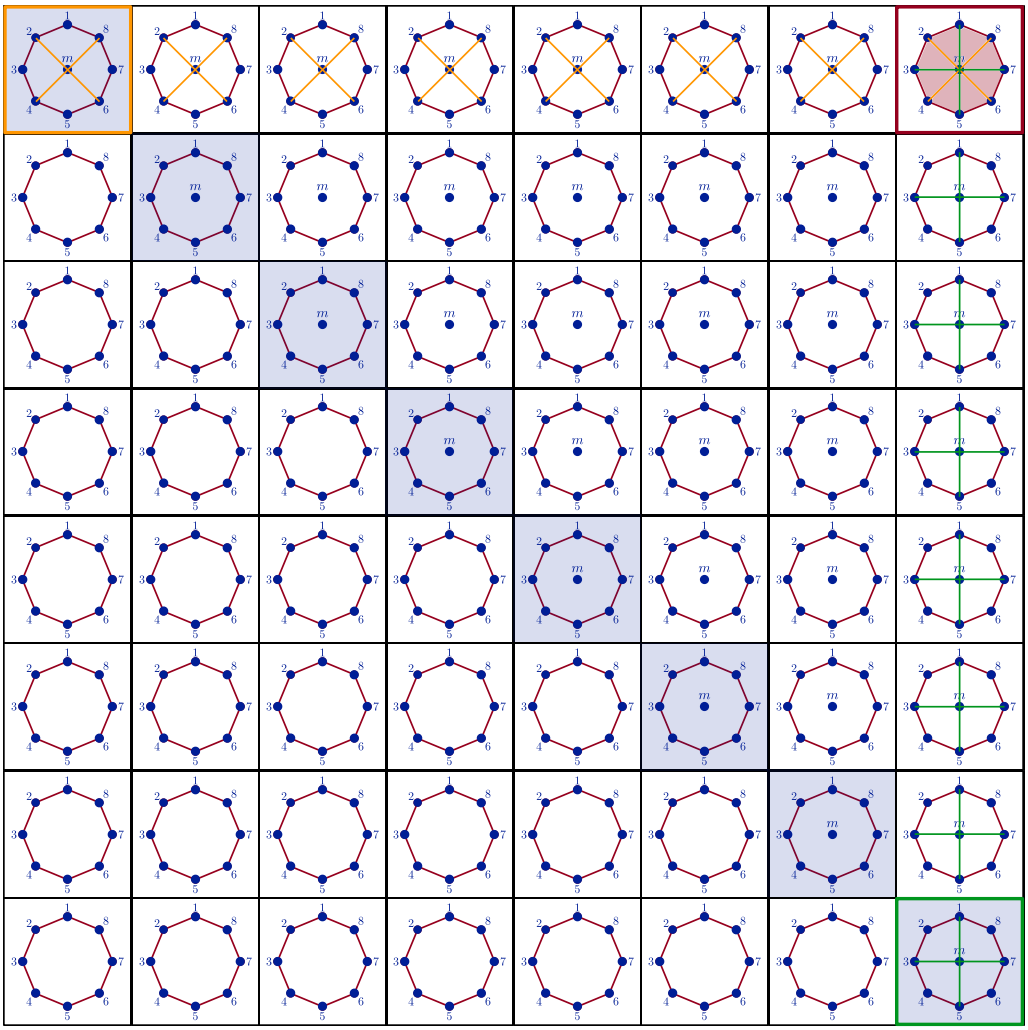


Figure 8 The *wheel* bifiltration consists of $\ell + 1$ 0-simplices, ℓ 1-simplices and ℓ 2-simplices, with $\ell = 8$ in this case. The only ℓ-critical 0-simplex is m entering the bifiltration along the blue staircase. This forces f_1^0 to map all generators of the orange (resp. green) 1-simplices to a generator of m with orange (resp. green) grade. The composition $f_1^0 \circ f_2^0$ then maps each generator of a purple 2-cell to the sum of a generator of m with orange grade and one with a green grade. The orange and green generators of m are connected by $\ell - 1$ relations. This yields $\ell - 1$ non-zero entries in each of the ℓ columns of $[h_i^0]$, making it dense.

5 Log-path algorithm

Shortcuts. In the preceding section, we leveraged the fact that the upset modules induced by simplices of a bifiltration admit path-shaped resolutions, allowing us to compute the correction terms f_i^1 and h_i^0 in a simple way. In the worst case, many generators and relations are mapped to long paths, making the matrices $[f_i^1]$ and $[h_i^0]$ dense ($\Omega(nk)$ size) and leading to a quadratic running time of $\Theta(nk)$ for the path algorithm.

Such long paths can be avoided by introducing shortcuts in the path resolution $\mathcal{P}_\sigma$ (4), as illustrated in Figure 9. Recall that the grades $\mathcal{G}(\sigma) = \{x_1, \cdots, x_{n_\sigma}\}$ generating $\mathrm{supp}(\sigma)$ are totally ordered (by their first coordinate). Now any two vertices $g_\sigma^{x_j}$ and $g_\sigma^{x_\ell}$ with $j < \ell$

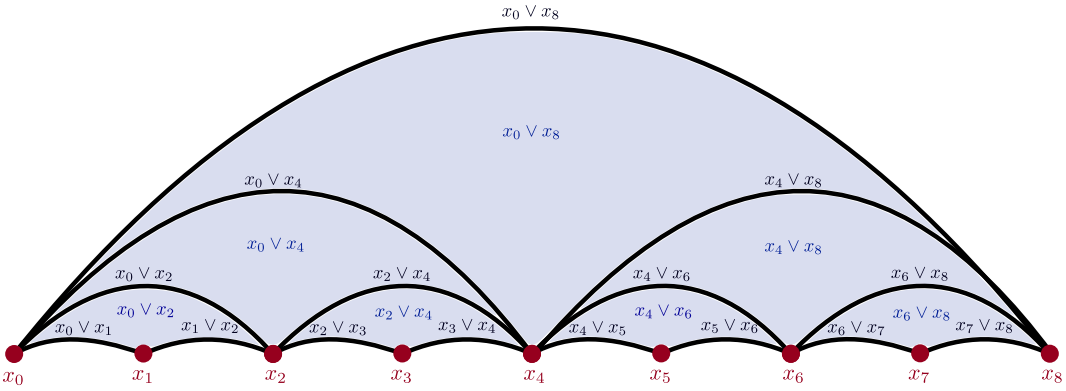


Figure 9 Log-path resolution labeled by the grades of its generators, relations and syzygies.

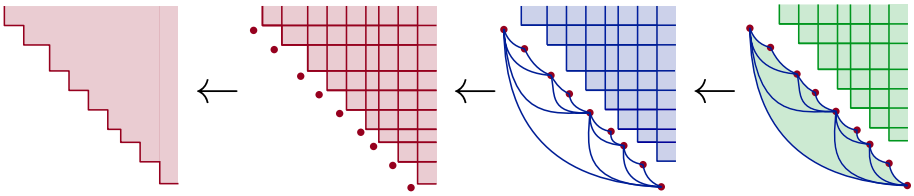


Figure 10 The new resolution of length 2 of an upset module U_σ.

are connected by an additional shortcut edge if there are numbers $s \geq 0, t > 0$ with $j = s2^t$ and $\ell = (s+1)2^t$. Extending $\mathcal{P}_\sigma$ by this edge corresponds to adding a relation r_σ^z to R_σ, where $p_\sigma(r_\sigma^z) = g_\sigma^{x_j} + g_\sigma^{x_\ell}$ and $z = x_j \vee x_\ell$. In total, only $O(n_\sigma)$ many edges are added.

This construction ensures that any two vertices $g_\sigma^{x_j}$ and $g_\sigma^{x_\ell}$ with $j < \ell$ are connected by a monotone path of length logarithmic in n_σ. A shortest monotone path can be constructed in a greedy way. We start in $g_\sigma^{x_j}$ and take the longest possible edge in each step that does not overshoot $g_\sigma^{x_\ell}$. We conclude the following extension of Lemma 1 (proved in [3, B.1]).

▶ **Lemma 4** (Log-path Lemma). *For any $g_\sigma^{x_j}$, $g_\sigma^{x_\ell}$ in G_σ, there exist $r_\sigma^{y_1}, \dots, r_\sigma^{y_m}$ in R_σ with $m = O(\log n_\sigma)$ such that $p_\sigma^1(r_\sigma^{y_1} + \cdots + r_\sigma^{y_m}) = g_\sigma^{x_j} + g_\sigma^{x_\ell}$. Moreover, the elements $r_\sigma^{y_1}, \dots, r_\sigma^{y_m}$ can be computed in $O(\log n_\sigma)$ time.*

Log-path resolutions. Lemma 4 appears to resolve the size issue: the same algorithm as in the previous section can be used, except that in the construction of h_i^0 and f_i^1 use Lemma 4 instead of Lemma 1. Then, the number of nonzero boundary coefficients for every generator and relation has only size $O(\log n_\sigma)$ instead of $O(n_\sigma)$, which would lead to an output boundary matrix with $O(n \log k)$ nonzero entries.

Now recall that a key reason for why the path construction in (7) yields a chain complex is the uniqueness of paths between vertices. This leads to the vanishing of the terms $h_i^0 \circ f_{i+1}^0 + f_{i-1}^1 \circ h_{i+1}^0$ and $h_i^0 \circ p_i^1 + f_{i-1}^1 \circ f_i^1$ when composing boundary maps in (7). However, after adding shortcut edges the paths between vertices are no longer unique. As illustrated in Figure 11, for a relation r_σ^y, the terms $h_i^0 \circ p_i^1(r_\sigma^y)$ and $f_{i-1}^1 \circ f_i^1(r_\sigma^y)$, when restricted to a component, may correspond to different paths connecting the same endpoints and may therefore enclose a cycle.

Any such cycle constitutes an obstruction to the vanishing of $h_i^0 \circ p_i^1 + f_{i-1}^1 \circ f_i^1$. To eliminate these obstructions, we fill the cycles with faces, or, in algebraic terms, introduce syzygies, thus extending the modified relations to a free resolution, as depicted in Figure 10:

$$0 \longleftarrow U_\sigma \xleftarrow{p_\sigma^0} G_\sigma \xleftarrow{p_\sigma^1} R_\sigma \xleftarrow{p_\sigma^2} S_\sigma \longleftarrow 0. \tag{8}$$

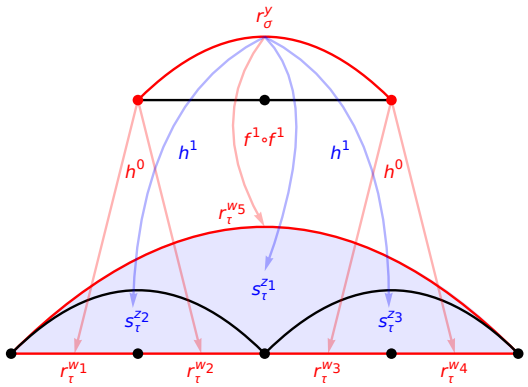


Figure 11 Cycle formed by paths $h^0 \circ p^1(r_\sigma^y)$ and $f^1 \circ f^1(r_\sigma^y)$ and its filling triangles $h^1(r_\sigma^y)$.

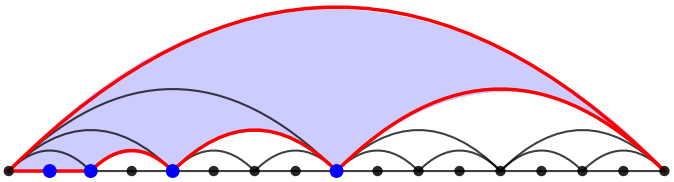


Figure 12 A simple red cycle in $\mathcal{L}$ for $k = 4$, together with the filling triangles in blue corresponding to the non-extremal vertices (with respect to the total order). Here each triangle $(x_{(s-1)2^t}, x_{s2^t}, x_{(s+1)2^t})$, with $s > 1$ odd, corresponds bijectively to its middle vertex x_{s2^t}.

The syzygies s_σ^z for S_σ correspond to the triangles with vertex grades $(x_{(s-1)2^t}, x_{s2^t}, x_{(s+1)2^t})$ for any odd $s > 0$; see Figure 9. The grade z is given by the grade of the longest edge, that is, $x_{(s-1)2^t} \vee x_{(s+1)2^t}$. The map p_σ^2 is defined by sending s_σ^z to the sum of the three edges of the triangle. We call (8) the *log-path resolution* of U_σ, noting that it is a bifiltered two-dimensional simplicial chain complex $\mathcal{L}_\sigma$. Its construction yields the following property:

▶ **Lemma 5.** *Let $r_\sigma^{y_1}, \ldots, r_\sigma^{y_l}$ be a cycle in R_σ. Then there exists a unique chain $s_\sigma^{z_1} + \cdots + s_\sigma^{z_q}$ of $q \leq l-2$ elements in S_σ such that $p_\sigma^2(s_\sigma^{z_1} + \cdots + s_\sigma^{z_q}) = r_\sigma^{y_1} + \cdots + r_\sigma^{y_l}$. Moreover, this chain can be computed in $O(l) = O(\log k)$ time.*

To fill a cycle in $\mathcal{L}_\sigma$, we decompose it into simple cycles and use the observation that a simple cycle is filled with the triangles corresponding to the non-extremal vertices x_{s2^t} of the cycle, see Figure 12. See B.2 of the full version [3] for details.

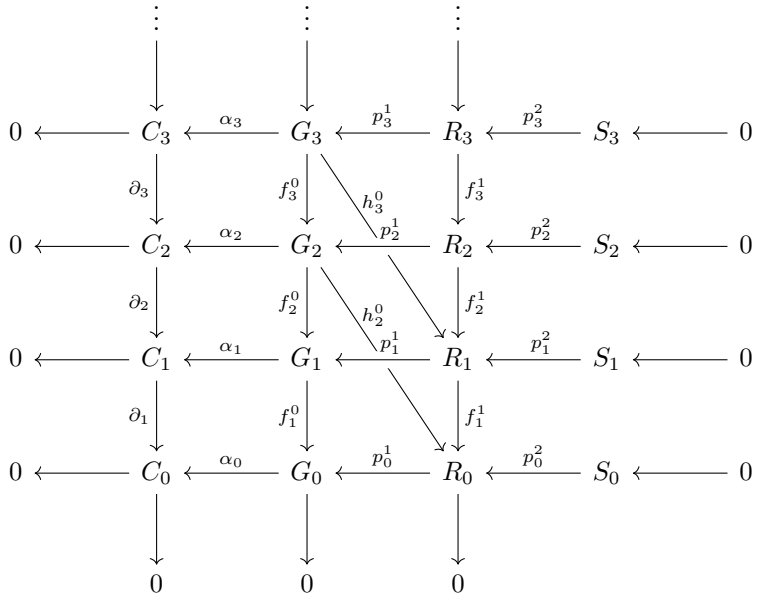
Log-path algorithm. We now describe the algorithm, again assuming that the input is a k-critical simplicial bifiltration of size n and constant dimension d with chain complex $C_\bullet$.

Step 1: Compute the log-path resolution $\mathcal{L}_i : G_i \xleftarrow{p_i^1} R_i \xleftarrow{p_i^2} S_i$ of each module C_i. This requires iterating through the generators of each input simplex and adding a total of $O(n)$ relations and syzygies, which takes $O(n)$ time in total.

Step 2: Compute $f_i^0 : G_i \to G_{i-1}$ in $O(n \log k)$ time as in Section 4.

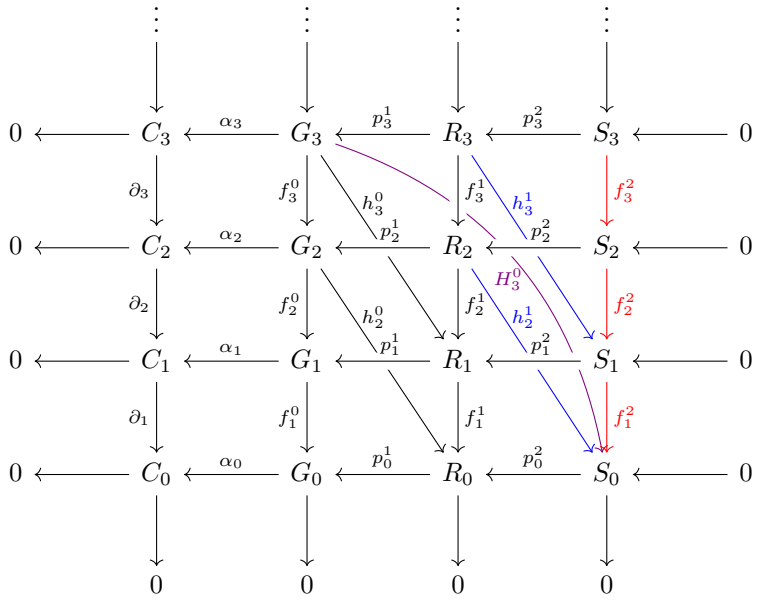
Step 3: Compute the maps $f_i^1 : R_i \to R_{i-1}$ and $h_i^0 : G_i \to R_{i-2}$ as in the path algorithm, except that paths between $g_\sigma^{x_j}, g_\sigma^{x_\ell}$ coming from the same simplex σ are computed via Lemma 4. We can find every path in $O(\log k)$ time and compute matrices $[f_i^1]$ and $[h_i^0]$ with at most $O(\log k)$ entries per column in a total running time of $O(n \log k)$ for this step.

Step 4. All maps computed in the previous steps yield the following extension of (7).



(9)

A construction as in Section 4 leads to boundary morphisms whose composition is non-zero. Again, the boundary defect is repaired by introducing correction maps f_i^2, h_i^1, and H_i^0:



(10)

Commutativity of Diagram (9) yields the following observation

▶ **Lemma 6.** *The following morphisms map to the kernel of* p_i^1*:*

$$f_{i+1}^1 \circ p_{i+1}^2, \quad f_{i+1}^1 \circ f_{i+2}^1 + h_{i+2}^0 \circ p_{i+2}^1, \quad \text{and} \quad h_{i+2}^0 \circ f_{i+3}^0 + f_{i+1}^1 \circ h_{i+3}^0. \tag{11}$$

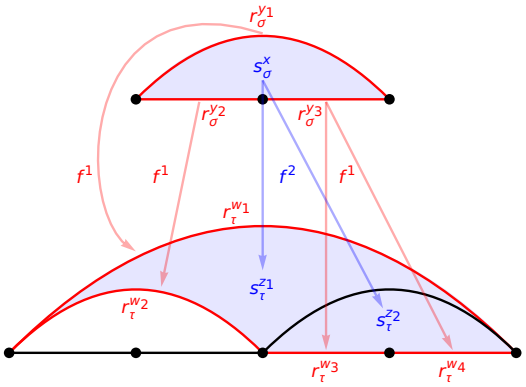


Figure 13 Cycle $f^1 \circ p^2(s_\sigma^x)$ and its filling triangles $f^2(s_\sigma^x)$.

The log-path resolution $\mathcal{L}_i$ of C_i is the direct sum of the simplicial chain complexes $\mathcal{L}_\sigma$, taken over each simplex σ in C_i. The map p_i^1 is given by the 1-dimensional boundary map of this complex, and hence its kernel is the collection of cycles in the 1-skeletons of $\mathcal{L}_\sigma$. More specifically, each triangle s_σ^z in S_{i+1} is sent by p_{i+1}^2 to its boundary, a sum of three edges.

By Lemma 6, applying f_{i+1}^1 to this boundary yields another cycle in R_i, which by construction of $\mathcal{L}_i$ decomposes as $\sum_\tau c_\tau$ for τ running over the facets of σ. For each such cycle c_τ, by Lemma 5 there is a unique chain b_τ in S_i such that $p_i^2(b_\tau) = c_\tau$, as illustrated in Figure 13. The map f_{i+1}^2 is then defined via $f_{i+1}^2(s_\sigma^z) := \sum_\tau b_\tau$, with τ running over the facets of σ. Thus f_{i+1}^2 repairs the boundary defect $f_{i+1}^1 \circ p_{i+1}^2$ as $f_{i+1}^1 \circ p_{i+1}^2 = p_i^2 \circ f_{i+1}^2$. Similar arguments lead to correction terms h_i^1, and H_i^0 satisfying

$$h_i^0 \circ p_i^1 + f_{i-1}^1 \circ f_i^1 = p_{i-2}^2 \circ h_i^1 \quad \text{and} \quad h_{i-1}^0 \circ f_i^0 + f_{i-2}^1 \circ h_i^0 = p_{i-3}^2 \circ H_i^0. \tag{12}$$

Their construction is analogous to the above, and we omit the details.

Note that in each of the three cases f_{i+1}^2, h_i^1, and H_i^0, the construction involves matrix multiplication (evaluation of the left hand sides in (12)), but also cycle filling (invoking Lemma 5). The former is efficient, as paths of logarithmic length lead to sparse matrices $[h_i^0]$ and $[f_i^1]$. More precisely, the column sparsity of $[f_i^1]$ and $[h_i^0]$ is $O(\log k)$, while in $[p_i^2]$ every column has exactly three non-zero entries. All sparse matrix products in Step 4 can thus be computed in $O(n \log^2 k)$ and the worst case column sparsity of the results is $O(\log^2 k)$. The resulting matrices contain cycles of length $O(\log^2 k)$, which can be filled by triangles in $O(\log^2 k)$ time. Overall, Step 4 takes $O(n \log^2 k)$ time.

We can now assemble the output chain complex as

$$0 \longleftarrow G_0 \xleftarrow{\begin{pmatrix} f_1^0 & p_0^1 \end{pmatrix}} G_1 \oplus R_0 \xleftarrow{\begin{pmatrix} f_2^0 & p_1^1 & 0 \\ h_2^0 & f_1^1 & p_0^2 \end{pmatrix}} G_2 \oplus R_1 \oplus S_0 \xleftarrow{\begin{pmatrix} f_3^0 & p_2^1 & 0 \\ h_3^0 & f_2^1 & p_1^2 \\ H_3^0 & h_2^1 & f_1^2 \end{pmatrix}} G_3 \oplus R_2 \oplus S_1 \longleftarrow \cdots \tag{13}$$

▶ **Theorem 7.** *The chain complex* (13) *is a free resolution of* $C_\bullet$.

For a proof, see chapters B.3 and B.4 of the full version [3]. We summarize our observations on the running time in the following theorem.

▶ **Theorem 8.** *A free resolution of the chain complex* $C_\bullet$ *induced by a* k*-critical bifiltration of description size* n *can be computed in* $O(n \log^2 k)$ *time and has* $O(n \log^2 k)$ *size.*

Applying this result to the modified wheel example ([3, A.3]) yields a non-minimal free resolution with smaller description size than any minimal one. Thus, perhaps surprisingly, minimal free resolutions are not asymptotically optimal in terms of description size.

6 Experimental evaluation

We implemented the path and log-path algorithm for computing a free resolution of a bifiltered chain complex in a C++ library called MULTI-CRITICAL [1]. The benchmark files are available upon request. The code expects a bifiltration given in scc2020 format [17], with the difference that for every simplex an arbitrarily long sequence of bigrades can be specified. The output is a chain complex in "proper" scc2020 format, representing a free filtered chain complex. Instead of outputting the result of our algorithms directly, the software can also post-process the filtered chain complex with the *multi-chunk* [16] library for minimizing the chain complex [14, 15].

All experiments were performed on a workstation with an Intel(R) Xeon(R) CPU E5-1650 v3 CPU (6 cores, 12 threads, 3.5GHz) and 64 GB RAM, running Ubuntu 16.04.5.

Test instances. We ran our experiments on a total of 342 test instances from different sources. First, we generated degree-Rips bifiltrations for (noisy) point samples drawn from a torus embedded in $\mathbb{R}^5$, from the "swiss roll" embedded in $\mathbb{R}^3$, and from the 7-dimensional unit cube, using the python package TADASETS. In all cases, we generated the complex up to 3-simplices.

Furthermore, we used a simple method for generating bifiltrations of a simplicial complex equipped with two non-negative real-valued functions f_0 and f_1 on its simplices, described in Figure 14. We employed this construction on meshes from the Aim@Shape repository[2], using the squared mean curvature and the distance to the barycenter as the two functions. Furthermore, we bifiltered the Delaunay triangulations from the same point samples as above (torus, cube, swiss roll) using the minimum enclosing radius of a simplex and the average distance to the $\lfloor \log n \rfloor$-nearest neighbors as filter functions. In all cases, we generated k-critical instances with $k = 2, 4, 8$. We computed the filtration values with CGAL [10, 26, 13].

Finally, we generated wheel bifiltrations as in Figure 8 for different sample sizes.

Comparison of path and log-path algorithms. Table 1 shows some of the results obtained from both algorithms on these datasets; the results for the other instances are similar.

In almost all cases, the log-path algorithm is slower than the path algorithm by a factor of up to 3 and produces an output complex of around twice the size. Also, it uses around twice as much memory (not shown in the table). Moreover, minimizing the chain complex (rightmost column of Table 1) further reduces the complex size significantly, generally at a small cost – sometimes it even saves time because the time used for minimization is less than the cost of producing the larger output file.

The only exception is the wheel bifiltration, where our experiments show the expected asymptotic worst-case behavior, with the log-path algorithm outperforming the path algorithm. In this case, minimizing the chain complex produced by the log-path algorithm will necessarily introduce a dense matrix and destroy the advantage of the log-path algorithm – for instance, its running time in the instance in the last row of Table 1 is 33 seconds, with 14.4 seconds for minimization itself and 18 seconds to write the 2.3 GB output file.

[1] Most recent version at `https://bitbucket.org/mkerber/multi_critical/src/main/`
[2] `http://visionair.ge.imati.cnr.it/`

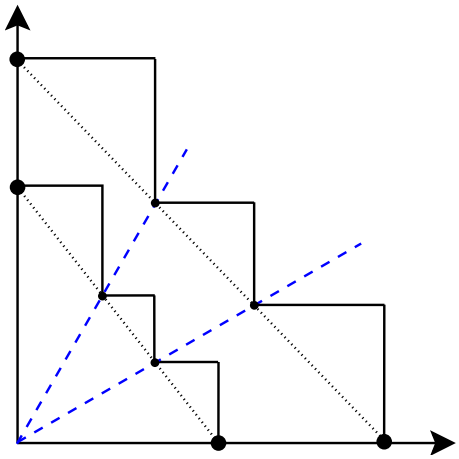


Figure 14 The figure shows the construction for two simplices σ and τ. The bigrades $(0, f_0(\sigma))$ and $(f_1(\sigma), 0)$ are connected by a line segment (dotted line), and the positive quadrant is equally dissected using $(k-2)$ lines through the origin (blue dashed lines, here $k = 4$). The intersection of dotted and dashed lines yield $(k-2)$ further bigrades for σ. The staircase for σ is drawn in solid. The same construction is done for τ; note that if $f_0(\sigma) \leq f_0(\tau)$ and $f_1(\sigma) \leq f_1(\tau)$, the staircase for σ is below the staircase for τ. In particular, the construction bifilters a simplicial complex if the functions f_0 and f_1 respect its face relations.

Computing minimal presentations. For a multi-critical filtered chain complex, we investigate the task of computing minimal presentation matrices for homology. We provide two variants: In the first variant, we compute a free resolution of the input chain complex with the path algorithm (which is usually faster than the log-path algorithm, as seen in the previous experiment) and subsequently minimize using MULTI-CHUNK. We then compute minimal presentations for each homology degree separately using the MPFREE library [18, 15].

The second variant operates by directly structuring the input complex into chain complex segments per degree. It then uses the Chacholski–Scolamiero–Vaccarino algorithm to compute a free implicit representation for each segment, before finally employing MPFREE once more to generate a minimal presentation.

The results of this experiment are presented in Table 2. For the degree-Rips instances, the first variant provides a modest improvement over the second. The reason is that the vast majority of simplices are in the top dimension 3, so that computing the presentation for H_2 is the bottleneck in the computation. This step, however, does not differ significantly in both approaches: most of the time is spent to determine which 3-simplices are killing 2-cycles, by reducing the boundary matrix for 2- and 3-simplices.

For the bifunction instances, the speed-up of the first variant using free resolutions is much more pronounced. The reason is that these instances have a more balanced distribution of simplices over different dimensions: while for the second variant, the algorithm for H_k still has to find the bounding $(k+1)$-simplices for every k, the multi-chunk algorithm makes use of the clearing optimization [9] and hence avoids the reduction of large parts of the boundary matrices. Remarkably, this technique is so effective that computing *all* minimal presentations via free resolutions is faster than computing a *single* minimal presentation via the approach using the Chacholski–Scolamiero-Vaccarino algorithm, even though restricting to a single dimension k allows this approach to disregard all chains in dimensions other than $k+1$, k, or $k-1$.

Table 1 Comparison of the path and the log-path algorithm. All times are in seconds. The "size" columns give the size of the input and output files. For deg-Rips and Delaunay, the numbers are averaged over 5 independently generated instances.

			log-path		path		path+multi_chunk	
Type	#points	size	time	size	time	size	time	size
deg-Rips torus in $\mathbb{R}^5$	32	10MB	6.59	84MB	2.44	42MB	2.33	18MB
	36	18MB	12.3	156MB	4.52	77MB	4.34	33MB
	42	37MB	27.0	355MB	9.73	171MB	9.52	73MB
	48	71MB	54.2	725MB	19.1	343MB	19.2	155MB
Delaunay swiss roll in $\mathbb{R}^3$ criticality 4	5000	257MB	33.4	521MB	19.7	389MB	16.3	32MB
	10000	534MB	69.8	1.1GB	40.4	823GB	35.0	63MB
	20000	1.1GB	142	2.2GB	83.7	1.7GB	78.7	131MB
	40000	2.2GB	297	4.7GB	175	3.5GB	191	282MB
Wheel	2500	184KB	0.06	440KB	0.78	31MB	0.78	30MB
	5000	376KB	0.13	932KB	2.91	132MB	3.39	120MB
	10000	804KB	0.26	2.0MB	13.5	574MB	11.7	526MB
	20000	1.7MB	0.61	4.0MB	49.4	2.3GB	53.1	2.3GB

Comparison with Macaulay2. While the Macaulay2 software includes a general `freeResolution` routine [25], it is unsuitable for our purposes. Its use requires converting our bigraded input into a graded chain complex over $\mathbb{Z}_2[x, y]$, a conversion that we found to be slow itself. More importantly, the core computation in Macaulay2 was orders of magnitude slower than our method, even on much smaller instances. We infer that the software was not designed for the large inputs typical of TDA. The conversion script is available on request.

7 Discussion

Our experimental evaluation suggests that the path algorithm often exhibits slightly better performance, which the log-path algorithm is more robust towards "bad" instances, with a relatively low overhead. Together, both variants contribute towards an efficient computational pipeline for multi-critical bifiltrations. Our results also complement recent development for computing degree-Rips bifiltrations more efficiently [2]. The finding that minimal chain complexes may necessitate a quadratic size in sparse matrix representations suggests that these representations may not be universally ideal for boundary matrices. Specifically, the matrix in Figure 8 admits a linear-size description, illustrating a gap in current approaches. It would be interesting to balance effective worst-case compression with efficient matrix processing, possibly by using an alternative data structure or by selective shortcutting.

Finally, our approach partially extends to simplicial complexes filtered with three parameters: a simplexwise minimal free resolution now has length 2, and suitable connecting maps compose into the same diagram as (10). Moreover, as described by Miller and Sturmfels [21], the simplexwise free resolution carries the structure of a planar graph, leading to a cubic-time algorithm. To break the cubic barrier, we will need to generalize the shortcut idea of Section 5 from paths to planar graphs, extending the free resolution of a simplex to length 3 (and introducing even more maps). We leave the details to future work.

Table 2 Computing all minimal presentations with the path algorithm and with the Chacholski–Scolamiero–Vaccarino algorithm. All running times are in seconds.

Type	#points	size	with path	with csv
deg-Rips cube in $\mathbb{R}^7$	32	8.7MB	5.66	6.22
	36	17MB	11.5	12.8
	42	35MB	28.7	34.0
	48	69MB	64.0	69.6
Delaunay torus in $\mathbb{R}^5$ criticality 8	140	129MB	14.7	79.7
	220	244MB	32.0	196
	340	435MB	56.8	403
	540	900MB	110	1017
hand, $k = 4$	34K	26MB	5.46	23.7
hand, $k = 8$	34K	55MB	13.1	55.3
hand, $k = 16$	34K	114MB	30.6	146
donna, $k = 2$	$503K$	170MB	44.9	131
donna, $k = 4$	503K	411MB	119	312
donna, $k = 8$	503K	895MB	282	860
pensatore, $k = 2$	998K	339MB	109	9585
pensatore, $k = 4$	998K	818MB	330	14054
pensatore, $k = 8$	998K	1.8GB	958	>4h

References

1 Ángel Javier Alonso. A Sparse Multicover Bifiltration of Linear Size. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:18, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2025.6.

2 Ángel Javier Alonso and Michael Kerber. deg_rips. https://bitbucket.org/mkerber/deg_rips/src/main/, 2026. Accessed: 2026-03-24.

3 Ulrich Bauer, Tamal K. Dey, Michael Kerber, Florian Russold, and Matthias Söls. Fast free resolutions of bifiltered chain complexes, 2026. arXiv:2512.08652.

4 Ulrich Bauer, Fabian Lenzen, and Michael Lesnick. Efficient Two-Parameter Persistence Computation via Cohomology. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:17, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2023.15.

5 Dave Bayer and Bernd Sturmfels. Cellular resolutions of monomial modules. *J. Reine Angew. Math.*, 502:123–140, 1998.

6 Nello Blaser, Morten Brun, Odin Hoff Gardaa, and Lars M. Salbu. Core bifiltration. *Journal of Applied and Computational Topology*, 9(30), 2025. doi:10.1007/s41468-025-00226-8.

7 Andrew J. Blumberg and Michael Lesnick. Stability of 2-parameter persistent homology. *Found. Comput. Math.*, 24(2):385–427, 2024. doi:10.1007/S10208-022-09576-6.

8 Wojciech Chachólski, Martina Scolamiero, and Francesco Vaccarino. Combinatorial presentation of multidimensional persistent homology. *Journal of Pure and Applied Algebra*, 221(5):1055–1075, 2017. doi:10.1016/j.jpaa.2016.09.001.

9 Chao Chen and Michael Kerber. Persistent homology computation with a twist. In *European Workshop on Computational Geometry (EuroCG)*, pages 197–200, 2011.

10 David Coeurjolly, Jaques-Olivier Lachaud, Konstantinos Katrioplas, Sébastien Loriot, Ivan Paden, Mael Rouxel-Labbé, Hossam Saeed, Jane Tournois, and Ilker O. Yaz. Polygon mesh processing. In *CGAL User and Reference Manual*. CGAL Editorial Board, 6.0.1 edition, 2024. URL: https://doc.cgal.org/6.0.1/Manual/packages.html#PkgPolygonMeshProcessing.

11 Tamal K. Dey and Florian Russold. Computing projective implicit representations from poset towers, 2025. doi:10.48550/arXiv.2505.08755.

12 Tamal K. Dey, Florian Russold, and Shreyas N. Samaga. Efficient Algorithms for Complexes of Persistence Modules with Applications. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 51:1–51:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.51.

13 Kaspar Fischer, Bernd Gärtner, Thomas Herrmann, Michael Hoffmann, and Sven Schönherr. Bounding volumes. In *CGAL User and Reference Manual*. CGAL Editorial Board, 6.0.1 edition, 2024. URL: https://doc.cgal.org/6.0.1/Manual/packages.html#PkgBoundingVolumes.

14 Ulderico Fugacci and Michael Kerber. Chunk Reduction for Multi-Parameter Persistent Homology. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2019.37.

15 Ulderico Fugacci, Michael Kerber, and Alexander Rolle. Compression for 2-parameter persistent homology. *Computational Geometry*, 109:101940, 2023. doi:10.1016/j.comgeo.2022.101940.

16 Michael Kerber. multi-chunk. https://bitbucket.org/mkerber/multi_chunk/src/master/, 2021. Accessed: 2025-12-02.

17 Michael Kerber and Michael Lesnick. The scc2020 format. https://bitbucket.org/mkerber/chain_complex_format/src/master/, 2021. Accessed: 2025-12-02.

18 Michael Kerber and Alexander Rolle. Fast minimal presentations of bi-graded persistence modules. In *2021 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 207–220, 2021. Software available at https://bitbucket.org/mkerber/mpfree/src/master/. doi:10.1137/1.9781611976472.16.

19 Michael Lesnick and Matthew Wright. Interactive visualization of 2-d persistence modules, 2015. arXiv:1512.00180.

20 Michael Lesnick and Matthew Wright. Computing minimal presentations and bigraded betti numbers of 2-parameter persistent homology. *SIAM Journal on Applied Algebra and Geometry*, 6(2):267–298, 2022. doi:10.1137/20M1388425.

21 Ezra Miller and Bernd Sturmfels. *Combinatorial Commutative Algebra*. Graduate Texts in Mathematics, 227. Springer New York, New York, NY, 2005.

22 Dmitriy Morozov and Luis Scoccola. Computing Betti tables and minimal presentations of zero-dimensional persistent homology. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry, SoCG 2025, Kanazawa, Japan, June 23-27, 2025*, volume 332 of *LIPIcs*, pages 69:1–69:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SOCG.2025.69.

23 Alexander Rolle. The Degree-Rips Complexes of an Annulus with Outliers. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 58:1–58:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2022.58.

24 Alexander Rolle and Luis Scoccola. Stable and consistent density-based clustering via multi-parameter persistence. *Journal of Machine Learning Research*, 25(258):1–74, 2024.

25 Gregory G. Smith and Mike Stillman. Complexes: beta testing new version of chain complexes. Version 1.0. A *Macaulay2* package available at https://github.com/Macaulay2/M2/tree/stable/M2/Macaulay2/packages. Accessed: 2025-12-02.

26 Hans Tangelder and Andreas Fabri. dD spatial searching. In *CGAL User and Reference Manual*. CGAL Editorial Board, 6.0.1 edition, 2024. URL: https://doc.cgal.org/6.0.1/Manual/packages.html#PkgSpatialSearchingD.

Locally Correct Interleavings Between Merge Trees

Thijs Beurskens
TU Eindhoven, The Netherlands

Tim Ophelders
TU Eindhoven, The Netherlands

Bettina Speckmann
TU Eindhoven, The Netherlands

Kevin Verbeek
TU Eindhoven, The Netherlands

Abstract

Merge trees are typically used as a topological summary of scalar fields. To analyze e.g. a time-varying scalar field via its merge tree representation, one needs a suitable method to match and compare two merge trees. We consider the interleaving distance as a method to match and compare merge trees. An *interleaving* between two merge trees consists of two maps, one in each direction. These maps must satisfy ancestor relations and hence introduce a "shift" between points and their image. An optimal interleaving minimizes the maximum shift; the interleaving distance is the value of this shift. However, to study the evolution of merge trees, we need not only a number but also a meaningful matching between the two trees. The two maps of an optimal interleaving induce a matching, but due to the bottleneck nature of the interleaving distance, this matching fails to capture local similarities between the trees. In this paper we hence propose a notion of local optimality for interleavings. To do so, we define the *residual interleaving distance*, a generalization of the interleaving distance that allows additional constraints on the maps. This allows us to define *locally correct interleavings*, which use a range of shifts across the two merge trees that reflect the local similarity well. We give a constructive proof that a locally correct interleaving always exists.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Interleaving distance, merge trees, local correctness, matchings, topological data analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.11

Related Version *Full Version*: `https://arxiv.org/abs/2512.16474`

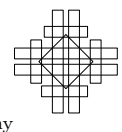
Funding *Thijs Beurskens*: Supported by the Dutch Research Council (NWO) under project no. OCENW.M20.089.
Tim Ophelders: Supported by the Dutch Research Council (NWO) under project no. VI.Veni.212.260.

1 Introduction

Terrains that vary over time are a common data type in various application areas, such as scientific computing or geographic information science. Typically such data sets are very large; to analyze them efficiently one often needs a compact abstraction that captures salient features. Topological data analysis (TDA) provides several *topological descriptors*, such as persistence diagrams, Reeb graphs, and Morse-Smale complexes, that can serve as abstractions for 2D, 3D, or higher-dimensional terrains. In this paper we focus in particular on *merge trees*, which are graph-based descriptors that encode the evolution of connected components of sublevel or superlevel sets of a terrain (see Figure 1). In the context of time-varying terrains, they have been used for feature tracking [7], cluster detection [16], and various other tasks in scientific visualization [17].


© Thijs Beurskens, Tim Ophelders, Bettina Speckmann, and Kevin Verbeek;
licensed under Creative Commons License CC-BY 4.0





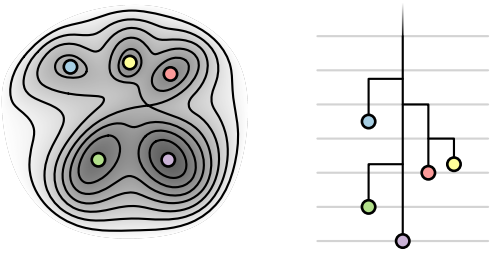


Figure 1 A terrain and its merge tree; leaves and internal vertices represent minima and saddles.

To analyze temporal sequences of terrains one does not only need a compact representation, but also efficient ways to compare these representations over time. For merge trees in particular, several distance measures have been proposed, including the (local) edit distance [13, 14], the Wasserstein distance [11], and the interleaving distance [8]. However, a distance measure returns only a numeric value that quantifies the similarity of two merge trees T_1 and T_2. In many application scenarios, we would also like to track the merge trees over time and hence need to understand which parts of T_1 most closely correspond to parts of T_2. That is, we are interested in a matching between T_1 and T_2.

The interleaving distance in fact relies on a type of matching between the two trees: the so-called *interleaving*. An interleaving is a pair of ancestor-preserving maps, one from T_1 to T_2 and one from T_2 to T_1, whose compositions send points to ancestors: if a point x of T_1 is mapped to a point y of T_2, then y must be mapped to an ancestor of x, and vice versa. The *interleaving distance* is the maximum *shift* – the distance between a point and its image – minimized over all interleavings between the trees. We give exact definitions of the interleaving distance and interleavings in Sections 2 and 3.

Two merge trees typically admit many optimal interleavings. The interleaving distance is a bottleneck measure and hence a given optimal interleaving might be locally quite "loose": the shift might exceed what is needed and thereby fail to capture local similarities between the trees; see Figure 2. In the following we are hence developing a notion of locally optimal interleavings which allows us to distinguish between "tight" and "loose" optimal interleavings.

Contributions. Our work is inspired by the concept of locally correct Fréchet matchings [3]; the *Fréchet distance* is a bottleneck measure for curves based on a matching between points on the curves. A locally correct Fréchet matching is then an optimal matching that remains optimal even when considering any two matched subcurves. This is a desirable feature, however, it cannot be directly reproduced for the interleaving distance. First of all, there might not even be a one-to-one correspondence between the points of T_1 and T_2. Furthermore,

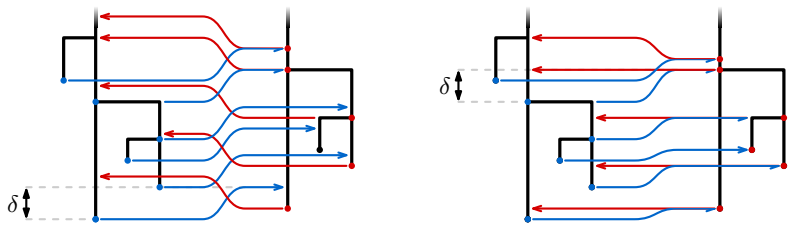


Figure 2 Two optimal interleavings between two merge trees (both with shift δ). Intuitively, the right interleaving induces a tighter and more meaningful matching than the left interleaving.

two optimal interleavings between different pairs of subtrees do not necessarily combine into an optimal interleaving between the two complete trees. As a first step we hence introduce a novel generalization of the interleaving distance that can handle additional constraints on the maps. Specifically, we aim to use constraints that force a point x to be mapped to an ancestor of a particular point y within the other tree. Over interleavings that satisfy such constraints, we aim to minimize the maximum shift over points x that are either unconstrained or map to a strict ancestor of their corresponding point y. We refer to the resulting distance as the *residual interleaving distance*. In Section 3, we provide formal definitions and show that an optimal interleaving exists for any constraints.

In Section 4 we then propose a definition for *locally correct interleavings* that builds upon the residual interleaving distance. Specifically, we define a restriction of an interleaving $\mathcal{I} = (\alpha, \beta)$ to two subsets S_1 of T_1 and S_2 of T_2 as the restriction of α to S_1 together with the restriction of β to S_2. We call $\mathcal{I}$ locally correct if it minimizes the residual interleaving distance with respect to any restriction of $\mathcal{I}$. We constructively prove that a locally correct interleaving always exists. All omitted proofs can be found in the full version.

Related work. The interleaving distance between merge trees was defined by Morozov, Beketayev, and Weber [8] as a tool to study the stability of merge trees. The authors of [1] establish a connection between the interleaving distance and the Gromov-Hausdorff distance, and they argue that approximating the interleaving distance within a factor of 3 is NP-hard. They also describe an $\mathcal{O}(\min(n, \sqrt{rn}))$-approximation algorithm, where n is the total size of the trees and r is the ratio of the longest to the shortest edge. Touli and Wang [15] provide an equivalent definition of the interleaving distance in terms of a single map and use this characterization to design an FPT-algorithm for computing the interleaving distance exactly. More recently, the authors of [6] show that the interleaving distance can be formulated in terms of a related metric [9] for labeled merge trees, also known as phylogenetic trees.

Several attempts have been made to develop versions of the interleaving distance that can be used in practical applications. In particular, the results in [6] gave rise to heuristic algorithms for computing the interleaving distance [4] and for comparing terrains [16, 18]. Pegoraro [10] uses yet another reformulation of the interleaving distance, in terms of couplings, as the basis of a heuristic for computing the distance via linear integer programming. Lastly, the authors of [2] introduce a variant of the interleaving distance that imposes total orders on the merge trees, which can be computed efficiently via a connection to the Fréchet distance between one-dimensional curves.

2 Preliminaries

Let T be a rooted tree, that is, a tree with one vertex identified as the *root*. In the remainder of this paper, we identify T with a *topological realization*: we represent each edge of T by a unit segment $[0, 1]$ and connect these segments according to the adjacencies of their corresponding edges in T. We distinguish elements of the topological realization T, which we refer to as *points* of T, from vertices of the combinatorial tree, which we denote by $V(T)$. A *merge tree* is a pair (T, f), where T is a finite rooted tree, and $f\colon T \to \mathbb{R} \cup \{\infty\}$ is a continuous *height function* on T that is strictly increasing towards the root and such that the root has height ∞. When clear from context, we use T to refer to the pair (T, f).

For two points x_1 and x_2 of T, we say x_1 is a *descendant* of x_2, written $x_1 \preceq x_2$, if there exists an f-monotonically increasing path in T from x_1 to x_2. If furthermore $x_1 \neq x_2$ then x_1 is a *strict descendant* of x_2. We say x_2 is a *(strict) ancestor* of x_1 if x_1 is a (strict) descendant of x_2. Lastly, for a point x and a height value $h \geq f(x)$, we use $x|^h$ to denote the unique ancestor of x at height $f(x|^h) = h$ (see Figure 3a).

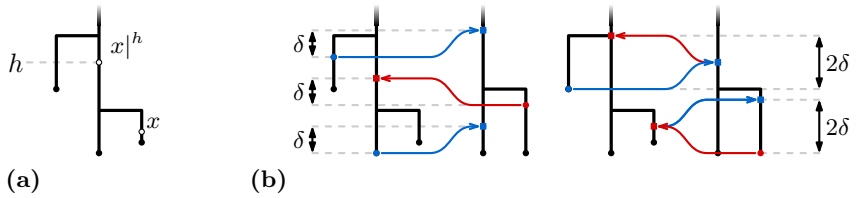


Figure 3 **(a)** The point $x|^h$ is the ancestor of x at height h. **(b)** A pair of δ-compatible maps. Left: **C1** and **C2** ensure that each point is mapped to a point of the other tree δ higher. Right: **C3** and **C4** ensure that both compositions map points to their ancestor 2δ higher.

Compatible maps. The original definition of the interleaving distance between two merge trees (T_1, f_1) and (T_2, f_2) was given by Morozov, Beketayev, and Weber [8].

▶ **Definition 1** ([8]). *Fix a value $\delta \geq 0$. Two continuous maps $\alpha\colon T_1 \to T_2$ and $\beta\colon T_2 \to T_1$ are* δ-compatible *between T_1 and T_2 if for all $x \in T_1$ and all $y \in T_2$:*

(C1) $f_2(\alpha(x)) = f_1(x) + \delta$,
(C2) $f_1(\beta(y)) = f_2(y) + \delta$,
(C3) $\beta(\alpha(x)) = x|^{f_1(x)+2\delta}$, *and*
(C4) $\alpha(\beta(y)) = y|^{f_2(y)+2\delta}$.

The interleaving distance $d(T_1, T_2)$ *between T_1 and T_2 is the infimum δ for which there exist δ-compatible maps (see Figure 3b).*

Critical values. It has been shown [1, 15] that the interleaving distance between any two merge trees attains a value in a set of *critical values*. Specifically, let $\Delta := \Delta(T_1, T_2)$ be the union of three sets Δ_1, Δ_2, and Δ_3, with

$$\begin{aligned} \Delta_1 &:= \{|f_1(v) - f_2(w)| : v \in V(T_1), w \in V(T_2)\}, \\ \Delta_2 &:= \{\tfrac{1}{2}|f_1(v_1) - f_1(v_2)| : v_1, v_2 \in V(T_1)\}, \\ \Delta_3 &:= \{\tfrac{1}{2}|f_2(w_1) - f_2(w_2)| : w_1, w_2 \in V(T_2)\}. \end{aligned} \tag{1}$$

▶ **Lemma 2** ([1, 15]). *For any two merge trees T_1 and T_2, it holds that $d(T_1, T_2) \in \Delta(T_1, T_2)$.*

3 Generalizing the interleaving distance

The interleaving distance is based on a pair of maps. As noted in [1], we can relax the requirements on these maps to allow varying height differences between points and their images. Building on this relaxed definition, we introduce a generalized version of the interleaving distance that accommodates constraints on the maps we consider. Specifically, each constraint specifies that a point x must be mapped to an ancestor of a particular point y within the other tree. Our goal is then to find the optimal pair of maps that "extends" a given set of constraints. We call the corresponding distance the *residual interleaving distance*. To define it formally, we first introduce *partial up-maps*, which represent the allowed maps between the two merge trees. In Section 3.1 we relax Definition 1 using partial up-maps. In Section 3.2 we then give the formal definition of the residual interleaving distance.

Partial up-maps. Let (T_1, f_1) and (T_2, f_2) be two merge trees. An *arrow* from T_1 to T_2 is any pair of points $(x, y) \in T_1 \times T_2$ such that $f_1(x) \leq f_2(y)$. The *shift* of an arrow (x, y) is the height difference between x and y, denoted $\sigma(x, y) := f_2(y) - f_1(x)$. Consider a subset of points $S \subseteq T_1$. A map $\varphi\colon S \to T_2$ is a *partial up-map* from S to T_2 if (i) $f_2(\varphi(x)) \geq f_1(x)$

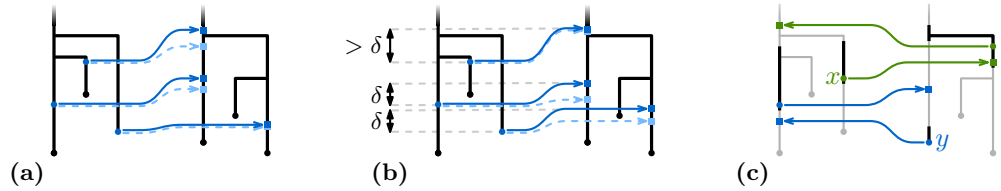


Figure 4 **(a)** A partial up-map (non-dashed) that extends another partial up-map (dashed). **(b)** A partial up-map φ (dashed) and the map $\varphi^{\uparrow}[\delta]$ (non-dashed). **(c)** A partial interleaving.

for all points x of S and (ii) it *preserves ancestors*, that is, $x_1 \preceq x_2$ implies $\varphi(x_1) \preceq \varphi(x_2)$ for all points x_1, x_2 of S. We use $\mathrm{dom}(\varphi) \coloneqq S$ to denote the *domain* of φ. For each point x of S, the pair $(x, \varphi(x))$ is an arrow. The *graph* of φ is the set of arrows of φ, denoted $A_\varphi \coloneqq \{(x, \varphi(x)) : x \in S\}$. We define the *shift* of the partial up-map φ as the supremum shift over all arrows of φ:

$$\sigma(\varphi) \coloneqq \sup\{\sigma(a) : a \in A_\varphi\}. \tag{2}$$

Given an arrow (x, y) from S to T_2, we say φ *extends* (x, y) if $\varphi(x) \succeq y$, and we say φ *uses* (x, y) if $\varphi(x) = y$. Similarly, for a subset $S' \subseteq S$ and a partial up-map φ' from S' to T_2, we say φ *extends* or *uses* φ' if, respectively, φ extends or uses all arrows of φ' (Figure 4a).

For any $\delta \geq 0$, we can define a partial up-map $\varphi^{\uparrow}[\delta]$ from S to T_2 that extends φ and such that the shift of each arrow is at least δ. Specifically, for each point x of S, let $h_x \coloneqq \max(f_1(x) + \delta, f_2(\varphi(x)))$ and define $\varphi^{\uparrow}[\delta](x) \coloneqq \varphi(x)|^{h_x}$ (see Figure 4b).

▶ **Lemma 3.** *For any $\delta \geq 0$, the map $\varphi^{\uparrow}[\delta]$ is a partial up-map that extends φ.*

Proof. Let $\varphi' \coloneqq \varphi^{\uparrow}[\delta]$. By construction, we have $f_2(\varphi'(x)) \geq f_1(x)$. To see that φ' preserves ancestors, take two points x_1, x_2 of S such that x_2 is an ancestor of x_1. Since φ preserves ancestors, $\varphi(x_2)$ is an ancestor of $\varphi(x_1)$. If $\varphi'(x_1) = \varphi(x_1)$, we get $\varphi'(x_1) \preceq \varphi(x_2) \preceq \varphi'(x_2)$. Otherwise $\varphi'(x_1)$ is a strict ancestor of $\varphi(x_1)$. By construction, $f_2(\varphi'(x_1)) = f_1(x_1) + \delta \leq f_1(x_2) + \delta \leq f_2(\varphi'(x_2))$. Both $\varphi'(x_1)$ and $\varphi'(x_2)$ are ancestors of $\varphi(x_1)$, so $\varphi'(x_2)$ is an ancestor of $\varphi'(x_1)$. So φ' is a partial up-map, and by construction also extends φ. ◀

We define arrows and up-maps from a subset S of T_2 to T_1 symmetrically.

Partial interleavings. Fix two subsets $S_1 \subseteq T_1$ and $S_2 \subseteq T_2$. A *partial interleaving* $\mathcal{P}$ between S_1 and S_2 is a pair of partial up-maps $\varphi\colon S_1 \to T_2$ and $\psi\colon S_2 \to T_1$ such that for all points x of S_1 and all points y of S_2 it holds that if $\varphi(x) \preceq y$ then $\psi(y) \succeq x$, and if $\psi(y) \preceq x$ then $\varphi(x) \succeq y$ (Figure 4c). The *shift* of $\mathcal{P}$, denoted $\sigma(\mathcal{P})$, is the maximum of the shift of φ and the shift of ψ. The *graph* of $\mathcal{P}$ is the union of the graph of φ and the graph of ψ, denoted $A_{\mathcal{P}} \coloneqq A_\varphi \cup A_\psi$. Given another partial interleaving $\mathcal{P}' = (\varphi', \psi')$, we say $\mathcal{P}$ *extends* $\mathcal{P}'$ if φ extends φ' and ψ extends ψ'. Similarly, $\mathcal{P}$ *uses* $\mathcal{P}'$ if φ uses φ' and ψ uses ψ'.

▶ **Lemma 4.** *For any $\delta \geq 0$, the pair $(\varphi^{\uparrow}[\delta], \psi^{\uparrow}[\delta])$ is a partial interleaving that extends (φ, ψ).*

Proof. Denote $\varphi' \coloneqq \varphi^{\uparrow}[\delta]$ and $\psi' \coloneqq \psi^{\uparrow}[\delta]$. By Lemma 3, the maps φ' and ψ' are partial up-maps that extend φ and ψ, respectively. Therefore, it suffices to show that the pair (φ', ψ') is a partial interleaving. Let x be a point of S_1 and y be a point of S_2 such that $\varphi'(x) \preceq y$. By construction, the point $\varphi'(x)$ is an ancestor of $\varphi(x)$, so y is also an ancestor of $\varphi(x)$. Since (φ, ψ) is a partial interleaving, the point $\psi(y)$ must be an ancestor of x. As $\psi'(y)$ is an ancestor of $\psi(y)$, we get $\psi'(y) \succeq x$. Symmetrically, if $\psi'(y) \preceq x$ then $\varphi'(x) \succeq y$. ◀

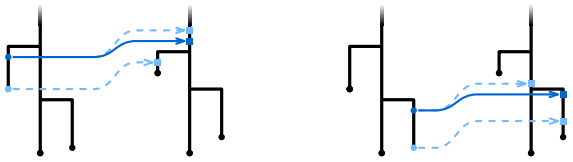


Figure 5 The shift of the dashed interleaving is not equal to a critical value, so we can obtain another interleaving (non-dashed) with strictly smaller shift by "pushing down" the image of each point. If we push below a vertex (right) we carefully need to choose the correct subtree.

3.1 Complete Interleavings

If $S_1 = T_1$, we call a partial up-map $\alpha\colon S_1 \to T_2$ a *complete up-map*, or simply an *up-map*, from T_1 to T_2. Symmetrically, if $S_2 = T_2$, we call a partial up-map $\beta\colon S_2 \to T_1$ a (complete) up-map from T_2 to T_1. Correspondingly, if $S_1 = T_1$ and $S_2 = T_2$, we call the partial interleaving (α, β) a *(complete) interleaving*. The defining property for a complete interleaving then becomes that for all $x \in T_1$ it holds that $\beta(\alpha(x)) \succeq x$ and for all $y \in T_2$ it holds that $\alpha(\beta(y)) \succeq y$.

▶ **Lemma 5.** *$d(T_1, T_2) = \inf\{\delta : T_1$ and T_2 admit a complete interleaving with shift $\delta\}$.*

Optimal interleavings. The interleaving distance is defined as an infimum, but in both [5, 10] the authors prove that the infimum can be replaced by a minimum: there always exists an interleaving with shift $d(T_1, T_2)$ (Theorem 6). We call an interleaving that realizes the interleaving distance an *optimal interleaving*. Both proofs rely on alternative definitions of the interleaving distance, namely in terms of labellings [5] and in terms of couplings [10]. We can show the same statement more directly: if the shift δ of an interleaving $\mathcal{I}$ is not equal to a critical value, we can obtain an interleaving whose shift $\delta' < \delta$ is equal to a critical value. Intuitively, we "push down" each arrow of $\mathcal{I}$ so that its shift becomes δ' (see Figure 5). Later, in Section 3.2, we describe a formal construction to prove a generalization of Theorem 6.

▶ **Theorem 6.** *Any two merge trees T_1 and T_2 admit an optimal interleaving.*

3.2 Residual Interleaving Distance

Fix a partial interleaving $\mathcal{P} = (\varphi, \psi)$. We refer to a (partial) interleaving that extends $\mathcal{P}$ as a (*partial*) *$\mathcal{P}$-extension*. We first observe that there always exists a complete $\mathcal{P}$-extension. Indeed, the maps that take every point of T_1 to the root at ∞ of T_2, and every point of T_2 to the root at ∞ of T_1 form a complete interleaving that extends $\mathcal{P}$.

Our goal is to find a "tightest possible" complete $\mathcal{P}$-extension, that is, an interleaving that extends $\mathcal{P}$ and minimizes the shift of the remaining arrows. For some intuition, consider the example in Figure 6. The map φ consists of a single (dashed) arrow (x, y). Intuitively, the map α_2 describes a "tighter" extension of φ than the map α_1. However, if we only disregard the shift of (x, y), the shift of the remaining arrows of α_1 is equal to the (supremum) shift of the remaining arrows of α_2. The arrow (x, y) implicitly induces a "fan" of arrows that all valid up-maps need to extend. Specifically, any ancestor of x needs to be mapped to an ancestor of y. To distinguish between the maps α_1 and α_2, we therefore not only disregard the shift of (x, y), but of any arrow that is "contained" within such a fan.

Formally, we define the *fan* of an arrow (x, y) as the set of arrows (x', y) for all ancestors x' of x with height at most that of y, written,

$$F[(x, y)] \coloneqq \{(x', y) : x' \in T_1 \text{ such that } x' \succeq x \text{ and } f_1(x') \leq f_2(y)\}.$$

The *fan* of a collection of arrows A, denoted $F[A]$, is the union of all fans $F[a]$ for $a \in A$. We refer to the fan $F[A_\varphi]$ as the *fan* of φ.

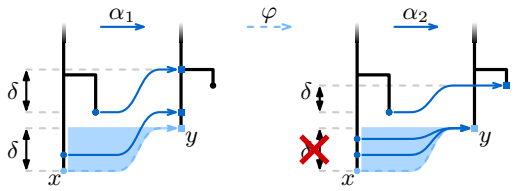


Figure 6 The φ-residual shift of arrows within the fan $F[(x, y)]$ (illustrated with the shaded area) is equal to 0, so the φ-residual shift of α_2 is smaller than the φ-residual shift of α_1.

We use fans to define the *φ-residual shift*, or *φ-shift* for short. Recall that we use A_φ to denote the set of all arrows $(x, \varphi(x))$. For an arrow a from T_1 to T_2, the φ-residual shift of a, denoted $\sigma_\varphi(a)$, is zero if a lies within $F[A_\varphi]$, and is $\sigma(a)$ otherwise. Similar to the definition of shift in Eq. 2, we define the φ-residual shift of an up-map φ', denoted $\sigma_\varphi(\varphi')$, as the supremum φ-residual shift over all arrows of φ'. That is,

$$\sigma_\varphi(x, y) \coloneqq \begin{cases} 0 & \text{if } (x, y) \in F[A_\varphi], \\ \sigma(x, y) & \text{else,} \end{cases} \qquad \sigma_\varphi(\varphi') \coloneqq \sup\{\sigma_\varphi(a) : a \in A_{\varphi'}\}.$$

Consider again the example in Figure 6. We disregard the shifts of all arrows within the fan $F[(x, y)]$. As a result, the φ-shift of α_2 is δ', whereas the φ-shift of α_1 is δ.

We define the *ψ-(residual) shift* of a partial up-map ψ' symmetrically. Finally, for a partial interleaving $\mathcal{P}' = (\varphi', \psi')$, we define the *$\mathcal{P}$-residual* shift, or *$\mathcal{P}$-shift*, of $\mathcal{P}'$ as the maximum of the φ-shift of φ' and the ψ-shift of ψ', denoted $\sigma_\mathcal{P}(\mathcal{P}')$.

▶ **Definition 7.** *The $\mathcal{P}$-residual interleaving distance $d_\mathcal{P}(T_1, T_2)$ between T_1 and T_2 is the infimum δ for which there exists a complete $\mathcal{P}$-extension whose $\mathcal{P}$-residual shift is δ.*

When they are clear from context, we omit the arguments T_1 and T_2 and simply write $d_\mathcal{P}$ to denote the $\mathcal{P}$-residual interleaving distance between T_1 and T_2. We say an extension is *optimal* if its residual shift is equal to the residual interleaving distance.

Critical values and pairs. For the remainder of this section, we consider *finite* partial interleavings $\mathcal{P}$: both $\mathrm{dom}(\varphi)$ and $\mathrm{dom}(\psi)$ consist of finitely many points. Recall from Section 2 that the non-residual interleaving distance always attains a value in the set of critical values $\Delta \coloneqq \Delta(T_1, T_2)$. For the residual interleaving distance, this set Δ is not always sufficient (see Figure 7). Therefore, we extend Δ to a set of $\mathcal{P}$-critical values. Specifically, we "mark" all vertices in T_1 and T_2, and all points x of T_1 and y of T_2 that appear in an arrow (x, y) or (y, x) of $\mathcal{P}$. We call these the *$\mathcal{P}$-critical points* of T_1 and T_2, denoted $C_1[\mathcal{P}]$ and $C_2[\mathcal{P}]$ respectively. Since $\mathcal{P}$ is finite, there are only finitely many critical points. We

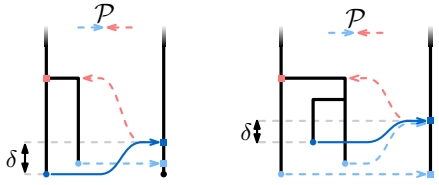


Figure 7 In both examples, the non-dashed arrow is (part of) an optimal interleaving; the value δ is not equal to a critical value. It is equal to a $\mathcal{P}$-critical value.

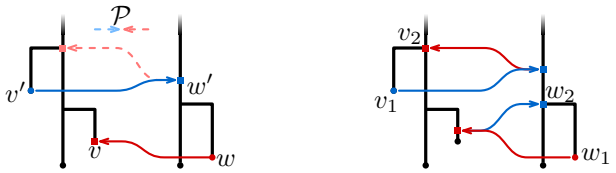


Figure 8 There are two types of critical pairs: arrow critical pairs (w, v) and (v', w') (left) and zigzag critical pairs (v_1, v_2) and (w_1, w_2) (right). The pair (v', w') correspond to a $\mathcal{P}$-critical value.

define a set $\Delta_1[\mathcal{P}]$ that contains all height differences between critical points:

$$\Delta_1[\mathcal{P}] \coloneqq \{|f_1(v) - f_2(w)| : v \in C_1[\mathcal{P}], w \in C_2[\mathcal{P}]\}.$$

We define the set of $\mathcal{P}$-*critical values* as the set $\Delta[\mathcal{P}](T_1, T_2) \coloneqq \Delta_1[\mathcal{P}] \cup \Delta_2 \cup \Delta_3$, where Δ_2 and Δ_3 are as defined in Eq. (1). Each critical value corresponds to a pair of critical points; we refer to such a pair b as a $\mathcal{P}$-*critical pair*. We distinguish two types; see Figure 8.

- If $b = (v, w)$ for a point $v \in C_1[\mathcal{P}]$ and a point $w \in C_2[\mathcal{P}]$, then b is an *arrow critical pair* that corresponds to the critical value $\sigma(v, w)$. We say a partial interleaving (φ, ψ) *uses* b if $\varphi(v) = w$. The case that $b = (w, v)$ is symmetric.
- If $b = (v_1, v_2)$ for two vertices v_1, v_2 in $V(T_1)$, then b is a *zigzag critical pair* that corresponds to the critical value $\frac{1}{2}(f_1(v_2) - f_1(v_1))$. We say a partial interleaving (φ, ψ) *uses* b if there is a point $y \in T_2$ with $f_2(y) = f_1(v_1) + \frac{1}{2}(f_1(v_2) - f_1(v_1))$ such that $\varphi(v_1) = y$ and $\psi(y) = v_2$. The case that $b = (w_1, w_2)$ for two vertices w_1, w_2 in $V(T_2)$ is symmetric.

Similar to Lemma 2, we show that the $\mathcal{P}$-residual interleaving distance always attains a value in the set of $\mathcal{P}$-critical values. A $\mathcal{P}$-critical pair is a *realizing* critical pair if it corresponds to the $\mathcal{P}$-residual interleaving distance. We can additionally show that there always exists an optimal $\mathcal{P}$-extension and that any such extension uses a realizing critical pair.

▶ **Lemma 8.** *Let $\mathcal{P}$ be a finite interleaving. For any complete $\mathcal{P}$-extension with $\mathcal{P}$-shift δ that does not use a critical pair that corresponds to δ, there exists another $\mathcal{P}$-extension whose $\mathcal{P}$-shift is equal to a $\mathcal{P}$-critical value that is strictly less than δ.*

Proof sketch. Consider a $\mathcal{P}$-extension $\mathcal{I} = (\alpha, \beta)$ with $\mathcal{P}$-shift δ, and assume $\mathcal{P}$ does not use a critical pair that corresponds to δ. We modify $\mathcal{I}$ into an extension $\mathcal{I}' = (\alpha', \beta')$ whose $\mathcal{P}$-shift is equal to a $\mathcal{P}$-critical value that is strictly less than δ. Let $\delta' \in \Delta[\mathcal{P}]$ be the greatest value that satisfies $\delta' < \delta$. Intuitively, we modify $\mathcal{I}$ by "pushing down" each arrow (recall Figure 5). We describe the construction of α'; the construction of β' is symmetric.

We assume, without loss of generality, that each arrow of α has shift at least δ'. Otherwise, we replace α by $\alpha^{\uparrow}[\delta']$. Since the residual shift of an arrow is bounded by its shift, and since $\delta' < \delta$, the φ-residual shift of $\alpha^{\uparrow}[\delta']$ is bounded by δ. For $x \in T_1$, let $v_x \in C_1[\mathcal{P}]$ be the highest descendant critical point of x; if $x \in C_1[\mathcal{P}]$, then $v_x = x$. Moreover, let $w_x \in C_2[\mathcal{P}]$ be the highest descendant critical point of $\alpha(v_x)$. Let $h_x \coloneqq \max(f_1(x) + \delta', f_2(w_x))$. We define $\alpha'(x)$ as the ancestor of w_x at height h_x, that is, $\alpha'(x) \coloneqq w_x|^{h_x}$.

It is not hard to prove that α' is an up-map that extends φ. To show that the φ-shift of α' is bounded by δ', we argue that any arrow (x, y) of α' with shift strictly greater than δ' must be inside the fan of φ. Specifically, we show that then also the shift δ^* of the arrow (v_x, w_x) is strictly greater than δ'. The pair (v_x, w_x) is a critical pair; since $\mathcal{I}$ does not use a critical pair that corresponds to δ, it moreover follows that δ^* cannot be equal to δ. So, by choice of δ', we get $\delta^* > \delta$. As a result, (v_x, w_x), and therefore also (x, w_x) must lie within the fan of φ: their residual shift is 0. Symmetrically, the map β' is an up-map that extends ψ and has ψ-shift at most δ'. We can then argue that the resulting pair $\mathcal{I}'$ is an interleaving. ◀

Lemma 8 directly implies Corollaries 9 and 10.

▶ **Corollary 9.** *If $\mathcal{P}$ is finite, then any optimal $\mathcal{P}$-extension uses a realizing critical pair.*

▶ **Corollary 10.** *If $\mathcal{P}$ is finite, then $d_{\mathcal{P}}(T_1, T_2) \in \Delta[\mathcal{P}](T_1, T_2)$.*

Let $\delta_1 \in \Delta[\mathcal{P}]$ be the greatest value that satisfies $\delta_1 \leq d_{\mathcal{P}}$ and let $\delta_2 \in \Delta[\mathcal{P}]$ be the smallest value that satisfies $\delta_2 > d_{\mathcal{P}}$. By definition of $d_{\mathcal{P}}$, for all $\varepsilon > 0$ there is a $\mathcal{P}$-extension with $\mathcal{P}$-shift $d_{\mathcal{P}} + \varepsilon$. In particular, for ε small enough, there is a $\mathcal{P}$-extension $\mathcal{I}$ with $\mathcal{P}$-shift $\delta_1 < \sigma_{\mathcal{P}}(\mathcal{I}) < \delta_2$. In other words, there are no critical pairs that correspond to $\sigma_{\mathcal{P}}(\mathcal{I})$. This means that $\mathcal{I}$ cannot use a critical pair that corresponds to the $\mathcal{P}$-shift of $\mathcal{I}$. By Lemma 8, it then follows that there exists a $\mathcal{P}$-extension with $\mathcal{P}$-shift $\delta_1 = d_{\mathcal{P}}$.

▶ **Theorem 11.** *Any finite partial interleaving $\mathcal{P}$ between any two merge trees T_1 and T_2 admits an optimal $\mathcal{P}$-extension.*

Isolated extensions. To reason about "greatest" shifts and "highest" ancestors, it is useful to have optimal extensions that satisfy two additional finiteness properties. Specifically, assume the $\mathcal{P}$-residual interleaving distance is strictly positive, that is, $d_{\mathcal{P}} > 0$. We say an optimal $\mathcal{P}$-extension $\mathcal{I}$ is *isolated* if (1) the $\mathcal{P}$-shift of only finitely many arrows of $\mathcal{I}$ is equal to $d_{\mathcal{P}}$, and (2) there exists some $\varepsilon > 0$ such that for each arrow (x, y) of $\mathcal{I}$ with $\mathcal{P}$-shift at least $d_{\mathcal{P}} - \varepsilon$, there exists a descendant x' of x, such that (x', y) is an arrow in $\mathcal{I}$ with the same target and $\mathcal{P}$-shift $d_{\mathcal{P}}$.

▶ **Lemma 12.** *If $\mathcal{P}$ is finite and $d_{\mathcal{P}} > 0$, there exists an isolated $\mathcal{P}$-extension.*

4 Locally correct interleavings

The interleaving distance is a bottleneck distance: it yields a single value that quantifies the worst discrepancy between the two merge trees. While this global value is useful for comparison, we are interested in the actual interleavings that realize the distance. Importantly, we seek interleavings that are not only globally optimal, but also locally meaningful. Specifically, if a part of the interleaving is fixed, we want the remainder to be optimal relative to this fixed part. To formalize this notion of local optimality, we build upon the residual interleaving distance introduced in the previous section.

Let (T_1, f_1) and (T_2, f_2) be two merge trees, and fix a partial interleaving $\mathcal{P} = (\varphi, \psi)$. For a subset $S_1 \subseteq \mathrm{dom}(\varphi)$, we define the *restriction* of φ to S_1 as the map $\varphi|_{S_1} \colon S_1 \to T_2$ given by $\varphi|_{S_1}(x) \coloneqq \varphi(x)$ for all points x of S_1. Any restriction of φ is a partial up-map, and φ extends all of its restrictions. The *restriction* of ψ to a subset $S_2 \subseteq T_2$ is defined symmetrically. Lastly, the pair $\mathcal{R} = (\varphi|_{S_1}, \psi|_{S_2})$ is a partial interleaving such that $\mathcal{P}$ extends $\mathcal{R}$. We call $\mathcal{R}$ a *restriction* of $\mathcal{P}$. Let $\mathcal{I}$ be a complete interleaving. We make the following observation:

▶ **Observation 13.** *For any restriction $\mathcal{R}$ of $\mathcal{I}$, it holds that $d_{\mathcal{R}}(T_1, T_2) \leq \sigma_{\mathcal{R}}(\mathcal{I})$.*

A complete interleaving $\mathcal{I}$ is *locally correct* if for all restrictions $\mathcal{R}$ of $\mathcal{I}$, the $\mathcal{R}$-residual shift of $\mathcal{I}$ is equal to the $\mathcal{R}$-residual interleaving distance.

▶ **Definition 14.** *A complete interleaving $\mathcal{I} = (\alpha, \beta)$ is* locally correct *if and only if for all $S_1 \subseteq T_1$ and $S_2 \subseteq T_2$, the restriction $\mathcal{R} = (\alpha|_{S_1}, \beta|_{S_2})$ satisfies*

$$d_{\mathcal{R}}(T_1, T_2) = \sigma_{\mathcal{R}}(\mathcal{I}). \tag{3}$$

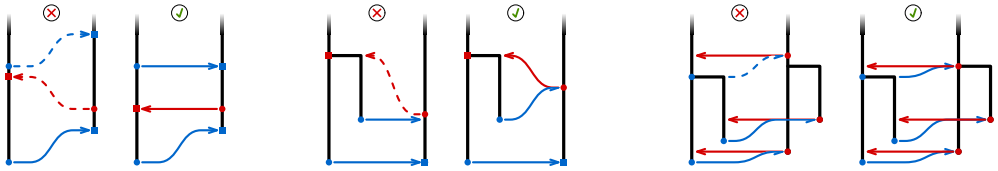


Figure 9 Three pairs of merge trees; for each pair we show an interleaving that is locally correct (✓), and one that is not (⊗). If we fix the non-dashed arrows in the ⊗ cases, the dashed arrows violate Eq. (3); e.g. for the middle example, if we fix the blue arrows, then the image of the red point needs to be an ancestor of both blue points, requiring a shift greater than the interleaving distance.

See Figure 9 for some examples of interleavings that are (not) locally correct.

Suppose $\mathcal{I}$ is locally correct. Taking $\mathcal{R}$ as the empty restriction, we see that $\mathcal{I}$ satisfies $d(T_1, T_2) = \sigma(\mathcal{I})$. In other words, any locally correct interleaving is an optimal interleaving. The reverse is not necessarily true; not every optimal interleaving is locally correct (see Figure 2). That raises the question: does a locally correct interleaving always exist? In the remainder of this section, we answer this question affirmatively.

▶ **Theorem 15.** *Any two merge trees T_1 and T_2 admit a locally correct interleaving.*

Overview of the proof. To prove Theorem 15, we give an explicit construction of a locally correct interleaving. Specifically, we incrementally build a partial interleaving $\mathcal{P}$ by augmenting it with "bottleneck" arrows. We show that with every augmentation, the $\mathcal{P}$-residual interleaving distance strictly decreases, until, after a finite number of iterations, it becomes zero. To conclude the proof, we then show that any complete interleaving that extends the resulting partial interleaving $\mathcal{P}$ is locally correct.

4.1 Bottlenecks

The interleaving distance is determined by the greatest shift among all arrows of an optimal interleaving. Locally, we might improve such an interleaving by "pushing down" some of the arrows, thereby decreasing their shifts. However, we cannot do this for all arrows; otherwise, we would obtain an interleaving whose shift is strictly smaller than the interleaving distance. We are interested in a *bottleneck*: a set of arrows that we cannot "push down".

We first define the *relative difference* of two, not necessarily related, partial up-maps φ and φ'. Specifically, consider the set of points for which either φ is defined and φ' is not, or for which both are defined but where they do not agree:

$$S \coloneqq \operatorname{dom}(\varphi) \setminus \{x \in \operatorname{dom}(\varphi) \cap \operatorname{dom}(\varphi') : \varphi(x) = \varphi'(x)\}.$$

We define the relative difference of φ and φ' as the restriction of φ to S, denoted $\varphi \setminus \varphi' \coloneqq \varphi|_S$. Similarly, for two partial interleavings $\mathcal{P} = (\varphi, \psi)$ and $\mathcal{P}' = (\varphi', \psi')$, we define the *relative difference* $\mathcal{P} \setminus \mathcal{P}'$ as the pair of relative differences $(\varphi \setminus \varphi', \psi \setminus \psi')$. The resulting pair is a restriction of $\mathcal{P}$, so it directly follows that any relative difference is a partial interleaving.

Augmentations. Fix a partial interleaving $\mathcal{P} = (\varphi, \psi)$. In the remainder of this section, we assume that $\mathcal{P}$ is finite, and that the $\mathcal{P}$-residual interleaving distance is strictly positive, that is, $d_{\mathcal{P}} > 0$. A $\mathcal{P}$-extension $\mathcal{Q}$ is a $\mathcal{P}$-*augmentation* if (i) the $\mathcal{P}$-residual shift of $\mathcal{Q}$ is at most $d_{\mathcal{P}}$, that is, $\sigma_{\mathcal{P}}(\mathcal{Q}) \leq d_{\mathcal{P}}$, and (ii) the $\mathcal{Q}$-residual interleaving distance is strictly less than $d_{\mathcal{P}}$, that is, $d_{\mathcal{Q}} < d_{\mathcal{P}}$. We say that $\mathcal{Q}$ is *minimal* if it does not extend any other $\mathcal{P}$-augmentation.

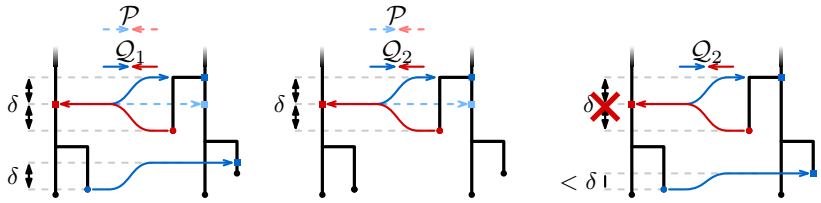


Figure 10 A partial interleaving $\mathcal{P}$ and two augmentations $\mathcal{Q}_1$ and $\mathcal{Q}_2$. $\mathcal{Q}_1$ is not a minimal augmentation, as $\mathcal{Q}_2$ is strictly smaller. The $\mathcal{Q}_2$-residual shift is strictly smaller than δ (right).

▶ **Lemma 16.** *Any finite interleaving $\mathcal{P}$ with $d_{\mathcal{P}} > 0$ admits a finite minimal $\mathcal{P}$-augmentation.*

Let $\mathcal{Q}$ be a minimal $\mathcal{P}$-augmentation. We refer to the relative difference $\mathcal{Q} \setminus \mathcal{P}$ as a $\mathcal{P}$-*bottleneck*. Intuitively, a bottleneck captures a minimal set of arrows that we need to "add" to decrease the residual interleaving distance. See Figure 10 for an illustration.

Recall (the proof of) Lemma 8. If we apply the lemma to a given $\mathcal{P}$-extension $\mathcal{I}$ that does not use a realizing critical pair, we can construct another $\mathcal{P}$-extension $\mathcal{I}'$ that has strictly smaller shift. The exact same proof still works for a slightly stronger statement: it suffices to assume that the relative difference $\mathcal{I} \setminus \mathcal{P}$ does not use a realizing critical pair.

▶ **Observation 17.** *Let $\mathcal{P}$ be a finite interleaving and let $\mathcal{I}$ be a $\mathcal{P}$-extension. If $\mathcal{I} \setminus \mathcal{P}$ does not use a realizing critical pair, there exists a $\mathcal{P}$-extension $\mathcal{I}'$ with $\sigma_{\mathcal{P}}(\mathcal{I}') < \sigma_{\mathcal{P}}(\mathcal{I})$.*

We use this observation to prove the following lemma.

▶ **Lemma 18.** *If $\mathcal{P}$ is finite and $d_{\mathcal{P}} > 0$, any $\mathcal{P}$-bottleneck $\mathcal{Q} \setminus \mathcal{P}$ uses a realizing critical pair.*

Proof. Let $\mathcal{I}$ be an optimal $\mathcal{Q}$-extension. Since $\mathcal{Q}$ extends $\mathcal{P}$, so does $\mathcal{I}$. Moreover, as $\mathcal{I}$ is optimal, the $\mathcal{Q}$-shift of $\mathcal{I}$ is equal to the $\mathcal{Q}$-residual interleaving distance, that is, $\sigma_{\mathcal{Q}}(\mathcal{I}) = d_{\mathcal{Q}}$. Since $\mathcal{Q}$ is an augmentation, the shift of all arrows of $\mathcal{Q} \setminus \mathcal{P}$ is at most $d_{\mathcal{P}}$, and $d_{\mathcal{Q}} < d_{\mathcal{P}}$. Combining, we obtain $\sigma_{\mathcal{P}}(\mathcal{I}) \leq d_{\mathcal{P}}$. In other words, $\mathcal{I}$ is an optimal $\mathcal{P}$-extension. By Observation 17, it follows that $\mathcal{I} \setminus \mathcal{P}$ uses a realizing critical pair b. As $d_{\mathcal{Q}} < d_{\mathcal{P}}$, we get that the $\mathcal{Q}$-shift of the arrows that correspond to b must be 0. In other words, $\mathcal{Q} \setminus \mathcal{P}$ uses b. ◀

A direct consequence of Lemma 18 is that if, for a given $\mathcal{P}$-extension $\mathcal{Q}$, the relative difference $\mathcal{Q} \setminus \mathcal{P}$ does not use a realizing critical pair, then $\mathcal{Q}$ cannot be a $\mathcal{P}$-augmentation. In other words, we have the following corollary:

▶ **Corollary 19.** *Let $\mathcal{P}$ be a finite interleaving with $d_{\mathcal{P}} > 0$, and let $\mathcal{Q}$ be a $\mathcal{P}$-extension. If $\sigma_{\mathcal{P}}(a) < d_{\mathcal{P}}(T_1, T_2)$ for all arrows $a \in A_{\mathcal{Q}}$, then $d_{\mathcal{Q}}(T_1, T_2) \geq d_{\mathcal{P}}(T_1, T_2)$.*

For a partial interleaving $\mathcal{P} = (\varphi, \psi)$, we use $\mathcal{P}.\varphi$ and $\mathcal{P}.\psi$ to denote φ and ψ, respectively.

4.2 Constructing a Locally Correct Interleaving

We show that a locally correct interleaving always exists by incrementally constructing one. Specifically, let $\mathcal{P}$ be an empty interleaving. We iteratively replace $\mathcal{P}$ with a finite minimal $\mathcal{P}$-augmentation, and we show that after a finite number of iterations the $\mathcal{P}$-residual interleaving distance becomes zero, at which point we stop iterating. We conclude by showing that for the resulting partial interleaving $\mathcal{P}$, any optimal $\mathcal{P}$-extension is locally correct. For this, we maintain three invariants throughout the entire construction. First, we argue that $\mathcal{P}$ remains *dominant*: the shift of each arrow of $\mathcal{P}$ is strictly greater than $d_{\mathcal{P}}$. Secondly, we

show that for all restrictions of $\mathcal{P}$ it holds that $d_{\mathcal{R}} \geq \sigma_{\mathcal{R}}(\mathcal{P})$; we say $\mathcal{P}$ is partially locally correct. Lastly, we show that after each step of the construction, $\mathcal{P}$ "specifies" at least one additional vertex of the input merge trees. Since each vertex can be specified at most once, it follows that the construction terminates.

Invariants. Formally, let $\mathcal{P}$ be a partial interleaving and assume $d_{\mathcal{P}} > 0$. Recall that another partial interleaving $\mathcal{Q}$ uses $\mathcal{P}$ if they completely agree on the domains of $\mathcal{P}.\varphi$ and $\mathcal{P}.\psi$.

▶ **Lemma 20.** *Let $\mathcal{P}$ be a dominant, finite interleaving with $d_{\mathcal{P}} > 0$. For any finite minimal $\mathcal{P}$-augmentation $\mathcal{Q}$, (i) $\mathcal{Q}$ uses $\mathcal{P}$, and (ii) $\mathcal{Q} \setminus \mathcal{P}$ uses only realizing critical pairs.*

We use this lemma to argue that $\mathcal{P}$ remains locally correct throughout the construction.

▶ **Lemma 21.** *Let $\mathcal{P}$ be a finite interleaving and suppose $d_{\mathcal{P}} > 0$. Let $\mathcal{Q}$ be a finite minimal $\mathcal{P}$-augmentation. If $\mathcal{P}$ is dominant and locally correct, then so is $\mathcal{Q}$.*

Proof. We first argue that $\mathcal{Q}$ is dominant. Let a be an arrow of $\mathcal{Q}$. If a is also an arrow of $\mathcal{P}$, then from the fact that $\mathcal{P}$ is dominant we know that $\sigma(a) > d_{\mathcal{P}} > d_{\mathcal{Q}}$. Otherwise, a must be an arrow of the bottleneck $\mathcal{Q} \setminus \mathcal{P}$. By Lemma 20, it then directly follows that $\sigma(a) = d_{\mathcal{P}} > d_{\mathcal{Q}}$.

Next, we argue that $\mathcal{Q}$ is locally correct. Consider a restriction $\mathcal{R}$ of $\mathcal{Q}$; we need to show that $d_{\mathcal{R}} \geq \sigma_{\mathcal{R}}(\mathcal{Q})$. If $\mathcal{R} = \mathcal{Q}$, we immediately obtain $\sigma_{\mathcal{R}}(\mathcal{Q}) = 0 \leq d_{\mathcal{R}}$. Otherwise, let $\mathcal{R}'$ be the restriction of $\mathcal{R}$ to the domains of $\mathcal{P}.\varphi$ and $\mathcal{P}.\psi$. Since $\mathcal{Q}$ uses $\mathcal{P}$, it follows that $\mathcal{R}'$ uses $\mathcal{P}$ and is hence a restriction of $\mathcal{P}$. If $\mathcal{R}' = \mathcal{P}$, then by definition of an augmentation we know that the $\mathcal{R}'$-shift of any arrow of $\mathcal{Q}$ is $d_{\mathcal{P}}$. We obtain $\sigma_{\mathcal{R}'}(\mathcal{Q}) = d_{\mathcal{P}}$. Moreover, since $\mathcal{R} \neq \mathcal{Q}$, the $\mathcal{R}$-shift of at least one arrow of $\mathcal{Q} \setminus \mathcal{P}$ is non-zero. We obtain $\sigma_{\mathcal{R}}(\mathcal{Q}) = \sigma_{\mathcal{R}'}(\mathcal{Q}) = d_{\mathcal{P}}$. Since $\mathcal{Q}$ is a minimal augmentation and extends $\mathcal{R}$, we know that $\mathcal{R}$ cannot be an augmentation of $\mathcal{P}$. It follows that $d_{\mathcal{R}} \geq d_{\mathcal{P}}$. Putting it together, we obtain $d_{\mathcal{R}} = \sigma_{\mathcal{R}}(\mathcal{Q})$.

Lastly, if $\mathcal{R}' \neq \mathcal{P}$, then since $\mathcal{P}$ is locally correct and $\mathcal{R}'$ is a restriction of $\mathcal{P}$, we have

$$d_{\mathcal{R}'} \geq \sigma_{\mathcal{R}'}(\mathcal{P}). \tag{4}$$

We first show that $\sigma_{\mathcal{R}'}(\mathcal{P}) \geq \sigma_{\mathcal{R}}(\mathcal{Q})$. First, none of the arrows of $\mathcal{R} \setminus \mathcal{R}'$ are arrows of $\mathcal{P}$, so $\sigma_{\mathcal{R}'}(\mathcal{P}) = \sigma_{\mathcal{R}}(\mathcal{P})$. Next, the $\mathcal{R}$-residual shift of $\mathcal{Q}$ is the maximum of the $\mathcal{R}$-residual shifts of $\mathcal{P}$ and $\mathcal{Q} \setminus \mathcal{P}$. Since $\mathcal{P}$ is dominant, we know that the shift of any arrow of $\mathcal{P}$ is strictly greater than $d_{\mathcal{P}}$. As $\mathcal{P} \setminus \mathcal{R}$ is not empty, it follows that $\sigma_{\mathcal{R}}(\mathcal{P}) > d_{\mathcal{P}}$. On the other hand, by definition of an augmentation, we know that the shift of any arrow of $\mathcal{Q} \setminus \mathcal{P}$ is at most $d_{\mathcal{P}}$. So, we get $\sigma_{\mathcal{R}}(\mathcal{Q}) = \max(\sigma_{\mathcal{R}}(\mathcal{P}), \sigma_{\mathcal{R}}(\mathcal{Q} \setminus \mathcal{P})) = \sigma_{\mathcal{R}}(\mathcal{P})$. Combining, we obtain $\sigma_{\mathcal{R}'}(\mathcal{P}) \geq \sigma_{\mathcal{R}}(\mathcal{Q})$.

Next, we argue that $d_{\mathcal{R}} \geq d_{\mathcal{R}'}$. By construction, $\mathcal{R}$ extends $\mathcal{R}'$. As observed before, we have $\sigma_{\mathcal{R}'}(\mathcal{P}) = \sigma_{\mathcal{R}}(\mathcal{P}) > d_{\mathcal{P}}$. Combining with Equation (4), we obtain $d_{\mathcal{R}'} \geq \sigma_{\mathcal{R}'}(\mathcal{P}) > d_{\mathcal{P}}$. Let a be an arrow of $\mathcal{R} \setminus \mathcal{R}'$. Then a is an arrow of the bottleneck $\mathcal{Q} \setminus \mathcal{P}$, so the shift of a is exactly $d_{\mathcal{P}}$. In other words, $\sigma_{\mathcal{R}'}(a) = d_{\mathcal{P}} < d_{\mathcal{R}'}$. This means that we can apply Corollary 19, for the finite interleaving $\mathcal{R}'$ and the $\mathcal{R}'$-extension $\mathcal{R}$, to obtain $d_{\mathcal{R}} \geq d_{\mathcal{R}'}$.

Thus, $d_{\mathcal{R}} \geq d_{\mathcal{R}'} \geq \sigma_{\mathcal{R}'}(\mathcal{P}) = \sigma_{\mathcal{R}}(\mathcal{Q})$, which concludes the proof. ◀

Recall that for each arrow (x, y), with $x \in T_1$ and $y \in T_2$, of $\mathcal{P}$, both x and y are critical points. However, only the point x has a "specified" target in the other tree. We say a point $x \in T_1$ or $y \in T_2$ is *specified* by $\mathcal{P}$, if $x \in \mathrm{dom}(\mathcal{P}.\varphi)$ or $y \in \mathrm{dom}(\mathcal{P}.\psi)$ respectively (see Figure 11). For our last invariant, we argue that $\mathcal{P}$ specifies all critical points that are not vertices, i.e., $C_1[\mathcal{P}] \setminus V(T_1) \subseteq \mathrm{dom}(\mathcal{P}.\varphi)$ and $C_2[\mathcal{P}] \setminus V(T_2) \subseteq \mathrm{dom}(\mathcal{P}.\psi)$.

▶ **Lemma 22.** *Let $\mathcal{P}$ be a finite interleaving and suppose $d_{\mathcal{P}} > 0$. Let $\mathcal{Q}$ be a finite minimal $\mathcal{P}$-augmentation. If $\mathcal{P}$ is dominant and specifies all critical points that are not vertices, then so does $\mathcal{Q}$. Moreover, $\mathcal{Q}$ specifies at least one vertex that $\mathcal{P}$ does not specify.*

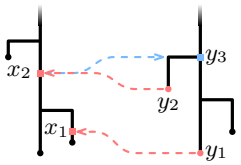


Figure 11 The points x_1, x_2, y_1, y_2 and y_3 are all critical, but only x_2, y_1 and y_2 are specified.

Proof. By Lemma 20, we know that the bottleneck $\mathcal{Q} \setminus \mathcal{P}$ uses only $\mathcal{P}$-critical pairs. Consider such a pair b. If b is an arrow critical pair, then without loss of generality assume $b = (x, y)$ for a point $x \in T_1$ and a point $y \in T_2$; the case $b = (y, x)$ is symmetric. By definition of an arrow critical pair, both x and y are $\mathcal{P}$-critical points. So the pair b does not create any additional $\mathcal{Q}$-critical points. As $\mathcal{P}$ already specifies all critical points that are not vertices and $\mathcal{Q}$ uses $\mathcal{P}$, we know that x must be a vertex. Since $\mathcal{P}$ is dominant, the shift of any arrow of $\mathcal{P}$ is strictly greater than $d_{\mathcal{P}}$. Moreover, the shift of any arrow of $\mathcal{Q} \setminus \mathcal{P}$ is exactly $d_{\mathcal{P}}$. Hence, the vertex x cannot be specified by $\mathcal{P}$: at least one new vertex is specified.

If b is a zigzag critical pair, then without loss of generality assume $b = (x_1, x_2)$ for two points x_1 and x_2 of T_1; if it is a pair of points of T_2 the argument is analogous. By definition of a zigzag critical pair, we know that both $x_1 \in V(T_1)$ and $x_2 \in V(T_1)$. Moreover, there is a point $y \in T_2$ such that $Q.\varphi(x_1) = y$ and $Q.\psi(y) = x_2$. So, x_1 is a vertex that is specified by $\mathcal{Q}$ but not by $\mathcal{P}$. Lastly, b creates a single $\mathcal{Q}$-critical point y, which is immediately specified. ◀

Finishing the proof of Theorem 15. Let $\mathcal{P}$ be the empty interleaving. Trivially, $\mathcal{P}$ is dominant and locally correct. Moreover, there are no (unspecified) critical points that are not vertices. Now, we iteratively replace $\mathcal{P}$ with a finite minimal $\mathcal{P}$-augmentation until the $\mathcal{P}$-residual interleaving distance becomes zero. By applying Lemmas 21 and 22 inductively, we maintain that $\mathcal{P}$ is dominant and locally correct, and specifies all critical points that are not vertices. Additionally, by Lemma 22, each iteration specifies at least one additional vertex. So after i iterations, at least i vertices are specified, and the total number of iterations is at most $|V(T_1)| + |V(T_2)|$. Afterwards, the residual interleaving distance is zero. Lemma 23 guarantees that a locally correct interleaving exists; its proof is similar to that of Lemma 21.

▶ **Lemma 23.** *Let $\mathcal{P}$ be a locally correct partial interleaving with $d_{\mathcal{P}} = 0$. Any optimal $\mathcal{P}$-extension is locally correct.*

Computing a locally correct interleaving. We sketch how the above construction can be translated into an incremental algorithm. To compute the residual interleaving distance, we modify the dynamic program by Touli and Wang [15] to also incorporate constraints induced by partial interleavings. We use the output to extract some augmentation. We then iteratively reduce the set of critical pairs by repeatedly calling the modified dynamic program until we obtain a (finite) minimal augmentation. This minimal augmentation, in turn, reduces the residual interleaving distance; we iterate until it has dropped to zero. To obtain a locally correct interleaving, we select suitable entries from the table filled by the modified dynamic program. A naive implementation of this algorithm takes n iterations, each of which needs to compute the residual interleaving distance $O(n^2)$ times to obtain a minimal augmentation (since an augmentation contains $O(n^2)$ critical pairs). So, the total running time of our algorithm is $O(n^3 T(n))$, where $T(n)$ is the running time of computing the residual interleaving distance. The algorithm by Touli and Wang [15] takes $O(n^4 2^\tau \tau^{\tau+2} \log n)$ time, where τ is a parameter related to the maximum degrees in the input trees.

5 Conclusion

We introduced locally correct interleavings between merge trees, which are optimal interleavings that are "tight". For this, we introduced the residual interleaving distance: a generalized version of the interleaving distance that can accommodate constraints on the interleavings. We presented a constructive proof that there always exists a locally correct interleaving.

We plan to further investigate the complexity of computing a locally correct interleaving, and we will study the effect of partial interleavings on the running time of the modified dynamic program. Furthermore, we will explore the effects of replacing the dynamic program with the approximation algorithm [1] or one of the variants or heuristics. Moreover, it would be interesting to study other (local) criteria of interleavings. Since locally correct interleavings are not necessarily unique, we could minimize their *total* shift, for a suitable definition of total. We could also consider a *lexicographic* optimization, inspired by lexicographic Fréchet matchings [12]. Lastly, we want to explore in which way locally correct interleavings can be translated to meaningful matchings between terrains.

References

1 P.K. Agarwal, K. Fox, A. Nath, A. Sidiropoulos, and Y. Wang. Computing the Gromov-Hausdorff distance for metric trees. *ACM Transactions on Algorithms*, 14(2):1–20, 2018. doi:10.1145/3185466.

2 T. Beurskens, T. Ophelders, B. Speckmann, and K. Verbeek. Relating interleaving and Fréchet distances via ordered merge trees. *Journal of Computational Geometry*, 17(1):1–36, 2026. doi:10.20382/jocg.v17i1a1.

3 K. Buchin, M. Buchin, W. Meulemans, and B. Speckmann. Locally correct Fréchet-matchings. *Computational Geometry*, 76(1):1–18, 2019. doi:10.1016/j.comgeo.2018.09.002.

4 J. Curry, H. Hang, W. Mio, T. Needham, and O. B. Okutan. Decorated merge trees for persistent topology. *Journal of Applied and Computational Topology*, 6(3):371–428, 2022. doi:10.1007/s41468-022-00089-3.

5 E. Gasparovic, E. Munch, S. Oudot, K. Turner, B. Wang, and Y. Wang. Intrinsic interleaving distance for merge trees. arXiv:1908.00063.

6 E. Gasparovic, E. Munch, S. Oudot, K. Turner, B. Wang, and Y. Wang. Intrinsic interleaving distance for merge trees. *La Matematica*, 4(1):40–65, 2025. doi:10.1007/s44007-024-00143-9.

7 W. Köpp and T. Weinkauf. Temporal merge tree maps: A topology-based static visualization for temporal scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1157–1167, 2022. doi:10.1109/TVCG.2022.3209387.

8 D. Morozov, K. Beketayev, and G. Weber. Interleaving distance between merge trees. Manuscript (accessed on 06-03-2025), 2013. URL: https://mrzv.org/publications/interleaving-distance-merge-trees/manuscript/.

9 E. Munch and A. Stefanou. The ℓ^∞-cophenetic metric for phylogenetic trees as an interleaving distance. In *Research in Data Science*, volume 17 of *Association for Women in Mathematics Series*, pages 109–127. Springer International Publishing, 2019. doi:10.1007/978-3-030-11566-1_5.

10 M. Pegoraro. A graph-matching formulation of the interleaving distance between merge trees. *AIMS Mathematics*, 10(6):13025–13081, 2025. doi:10.3934/math.2025586.

11 M. Pont, J. Vidal, J. Delon, and J. Tierny. Wasserstein distances, geodesics and barycenters of merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):291–301, 2022. doi:10.1109/TVCG.2021.3114839.

12 G. Rote. Lexicographic Fréchet matchings. In *Proc. 30th European Workshop on Computational Geometry (EuroCG'14)*, 2014. doi:10.17169/refubium-19845.

13 R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan. Edit distance between merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1518–1531, 2020. doi:10.1109/TVCG.2018.2873612.

14 R. Sridharamurthy and V. Natarajan. Comparative analysis of merge trees using local tree edit distance. *IEEE Transactions on Visualization and Computer Graphics*, 29(2):1518–1530, 2021. doi:10.1109/TVCG.2021.3122176.

15 E.F. Touli and Y. Wang. FPT-algorithms for computing the Gromov-Hausdorff and interleaving distances between trees. *Journal of Computational Geometry*, 13:89–124, 2022. doi:10.20382/jocg.v13i1a4.

16 L. Yan, T. B. Masood, F. Rasheed, I. Hotz, and B. Wang. Geometry aware merge tree comparisons for time-varying data with interleaving distances. *IEEE Transactions on Visualization and Computer Graphics*, 29(8):3489–3506, 2022. doi:10.1109/TVCG.2022.3163349.

17 L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Computer Graphics Forum*, 40(3):599–633, 2021. doi:10.1111/cgf.14331.

18 L. Yan, Y. Wang, E. Munch, E. Gasparovich, and B. Wang. A structural average of labeled merge trees for uncertainty visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):832–842, 2020. doi:10.1109/TVCG.2019.2934242.

Dynamic and Streaming Algorithms for Union Volume Estimation

Sujoy Bhore
Indian Institute of Technology Bombay, Mumbai, India

Karl Bringmann
Saarland University and Max-Planck-Institute for Informatics, Saarbrücken, Germany

Timothy M. Chan
University of Illinois at Urbana-Champaign, IL, USA

Yanheng Wang
Saarland University and Max-Planck-Institute for Informatics, Saarbrücken, Germany

Abstract

The *union volume estimation* problem asks to $(1 \pm \varepsilon)$-approximate the volume of the union of n given objects $X_1, \ldots, X_n \subset \mathbb{R}^d$. In their seminal work in 1989, Karp, Luby, and Madras solved this problem in time $O(n/\varepsilon^2)$ in an oracle model where each object X_i can be accessed via three types of queries: obtain the volume of X_i, sample a random point from X_i, and test whether X_i contains a given point x. This running time was recently shown to be optimal [Bringmann, Larsen, Nusser, Rotenberg, and Wang, SoCG'25]. In another line of work, Meel, Vinodchandran, and Chakraborty [PODS'21] designed algorithms that read the objects in one pass using polylogarithmic time per object and polylogarithmic space; this can be phrased as a dynamic algorithm supporting insertions of objects for union volume estimation in the oracle model.

In this paper, we study algorithms for union volume estimation in the oracle model that support both *insertions* and *deletions* of objects. We obtain the following results:

1. an algorithm supporting insertions and deletions in polylogarithmic update and query time and linear space (this is the first such dynamic algorithm, even for 2D triangles);
2. an algorithm supporting insertions and suffix queries (which generalizes the sliding window setting) in polylogarithmic update and query time and space;
3. an algorithm supporting insertions and deletions of convex bodies of constant dimension in polylogarithmic update and query time and space.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Data structures design and analysis

Keywords and phrases union volume estimation, dynamic algorithms, streaming algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.12

Related Version *Full Version*: `https://arxiv.org/pdf/2602.16306` [5]

Funding *Karl Bringmann and Yanheng Wang:* This work is part of the project TIPEA that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 850979).

1 Introduction

In the *union volume estimation* problem, we are given *objects* $X_1, \ldots, X_n$, which are assumed to be measurable sets in $\mathbb{R}^d$, and the task is to $(1 \pm \varepsilon)$-approximate the volume of $\bigcup_{i=1}^n X_i$.[1] This is a fundamental problem in computer science, with connections to several areas such as

[1] A discrete variant of the problem, estimating the *cardinality* of the union of sets $X_1, \ldots, X_n \subset \mathbb{Z}^d$, can be easily reduced to the continuous version, by replacing each grid point with a unit grid cell.

© Sujoy Bhore, Karl Bringmann, Timothy M. Chan, and Yanheng Wang;
licensed under Creative Commons License CC-BY 4.0

network reliability [32, 34], DNF counting [34], general model counting [45, 14], probabilistic databases [35, 21, 46], and probabilistic query evaluation on databases [13, 12]. Perhaps the most popular special case in computational geometry is *Klee's measure problem* [36], where each object is a d-dimensional axis-aligned box. There has been extensive work on Klee's measure problem (either exact or approximate) in constant dimensions [44, 15, 16, 37, 27, 10].

To capture a wide variety of input objects with a unified framework, Karp and Luby [33] introduced an *oracle model*. An algorithm in this model can access each object X_i exclusively via three types of queries: (i) X_i.SIZE() gives the volume of X_i, (ii) X_i.SAMPLE() samples a point uniformly at random from the set X_i, and (iii) X_i.CONTAINS(x) tests whether a given point x is contained in X_i. The time complexity of the algorithm is measured by the number of queries plus the number of additional steps; that is, each query is assumed to take unit time. In a seminal result, Karp, Luby, and Madras [34] gave an algorithm in the oracle model that solves union volume estimation in time $O(n/\varepsilon^2)$ with constant success probability, where ε is the approximation precision. Black-box usage of this result also yields the best known algorithms for several applications, including DNF counting. (For Klee's measure problem, it implies an $O(nd/\varepsilon^2)$-time approximation algorithm [9]; in small dimensions, this was improved to $O((n + 1/\varepsilon^2)\log^{O(d)} n)$ time at last year's SoCG [10].) Only recently has a matching lower bound been established: $\Omega(n/\varepsilon^2)$ queries are necessary to solve union volume estimation in the oracle model [10].

With the static setting now fully understood, attention naturally shifts to the dynamic setting, where the input evolves over time through insertions and deletions of objects. The study of *dynamic union volume estimation* has a twofold motivation. On one hand, it captures modern data-driven applications where datasets are inherently mutable, and volume estimates must be maintained continuously rather than recomputed from scratch after each update. On the other hand, the dynamic setting presents a compelling theoretical challenge, as it is far from obvious whether the tools developed for the static setting can be dynamized efficiently. Indeed, dynamic geometric problems have repeatedly proven to be substantially more difficult than their static counterparts. An extensive body of work in recent years deals with dynamic geometric problems such as set cover [2, 17], independent set [28, 7, 6], planar point location [43], connectivity [19, 18], and nearest neighbor search [1], yet in this broader context dynamic union volume estimation remains largely unexplored. The following question has not been resolved even for very simple classes of objects such as triangles in $\mathbb{R}^2$:

Q1: Can we solve dynamic union volume estimation in the oracle model under insertions and deletions of objects in $\text{poly}\left(\log n, \frac{1}{\varepsilon}\right)$ *update and query time?*

An even more demanding paradigm is the (one-pass) streaming setting, where the algorithm has to not only handle updates, but use space that is typically required to be polylogarithmic in the input size. Earlier work has explored streaming algorithms under strong restrictions on the input objects. For example, Tirthapura and Woodruff [47] designed a dynamic streaming algorithm for Klee's measure problem on axis-aligned boxes. However, the problem remains unaddressed for more general classes of objects:

Q2: Can we solve dynamic union volume estimation in the oracle model under insertions and deletions of objects in $\text{poly}\left(\log n, \frac{1}{\varepsilon}\right)$ *space as well as update and query time?*

In the restricted *insertion-only* setting, Meel, Vinodchandran, and Chakraborty [40] gave a positive answer to this question, using $O\left(\frac{1}{\varepsilon^2}\log^2(n)(\log\log(n) + \log(\frac{1}{\varepsilon}))\right)$ time per update and query, and $O\left(\frac{1}{\varepsilon^2}\log(n)\right)$ space in the oracle model.[2] However, their algorithm cannot handle deletions.

[2] In this paper, we assume that points from the input objects and $O(\log n)$-bit integers fit into a machine

2 Our contributions

2.1 Dynamic algorithm with insertions and deletions

We answer our driving question (Q1) positively, not only for 2D triangles, 3D simplices, and other common types of objects encountered in computational geometry, but in the full generality of the oracle model. Our new dynamic algorithm can handle arbitrary sequences of both insertions and deletions.

More precisely, we consider an initially empty multiset of objects $\mathcal{X}$ under the updates INSERT(X), which adds object X to $\mathcal{X}$, and DELETE(X), which removes object X from $\mathcal{X}$. Upon query ESTIMATE(), the algorithm outputs a $(1 \pm \varepsilon)$-approximation to the volume $\mathrm{vol}\left(\bigcup_{X \in \mathcal{X}} X\right)$.

During insertion/deletion the algorithm is given oracle access to the inserted/deleted object. We also need the ability to *store* objects (or their oracles), in case of future lookup of the previously inserted objects. We make the natural assumption that (i) the argument of INSERT(X) and DELETE(X) is a *pointer* to the oracle of X, (ii) we can store a pointer in a machine word, and (iii) knowing the pointer allows us to access the oracle to which the pointer is pointing. In this interpretation, $\mathcal{X}$ is treated as a multiset of pointers: the same pointer can be inserted multiple times, and upon deletion, the multiplicity of the pointer is decreased by one. Naturally, the multiplicity of any pointer must be non-negative throughout the process (in particular, objects being deleted must have previously been inserted).

▶ **Theorem 1.** *There is a dynamic algorithm in the oracle model that maintains a multiset $\mathcal{X}$ of objects under updates* INSERT(X) *and* DELETE(X)*. Upon query* ESTIMATE()*, it outputs a $(1 \pm \varepsilon)$-approximation to* $\mathrm{vol}\left(\bigcup_{X \in \mathcal{X}} X\right)$ *with high probability. The amortized expected update time is $O(\log^5(n/\varepsilon)/\varepsilon^2)$, the query time is $O(\log^4(n)/\varepsilon^2)$, and the space complexity is $O(n + \log^4(n)/\varepsilon^2)$.*

Note that since the algorithm operates in the general oracle model, it works well even for *high-dimensional* objects (e.g., for simplices in $\mathbb{R}^d$, with polynomial dependence on d).

2.2 Streaming algorithm with insertions and suffix queries

In our remaining results, we turn to streaming algorithms and our driving question (Q2) and show that polylogarithmic time *and* space can be achieved in certain settings. We first consider the *sliding window* setting [22], which supports insertions of objects and querying over the last w objects for a fixed window size w; thus, objects are deleted in the same order as they are inserted. Numerous results and techniques have been developed in the sliding window setting (see [41, 3, 8] for an overview and [48, 30, 25] for recent work), but no such result was known for union volume estimation.

We obtain the first efficient sliding-window algorithm for union volume estimation. Again, it works in the general oracle model! Moreover, the result holds in an extension of the sliding-window setting, where at each time $t = 1, \ldots, n$ an object X_t is inserted (but cannot be deleted), and a query ESTIMATE(s) may return a $(1 \pm \varepsilon)$-approximation to the volume of the *suffix* $\mathrm{vol}(X_s \cup \cdots \cup X_t)$ with high probability for *any* number s the user specifies.

word, and space complexity counts the number of machine words. An arithmetic operation on machine words is assumed to take unit time. Both are natural assumptions for a continuous universe. Some previous work such as [40] focuses on sets from a finite universe and studies bit complexity; when stating previous work in the introduction, we have translated their results into our setting. In the finite universe Ω, different trade-offs between $\log(n)$ and $\log|\Omega|$ were obtained in [39, 42].

▶ **Theorem 2.** *There is an algorithm that implicitly maintains a sequence of objects $X_1, \ldots, X_t$ of aspect ratio M. The algorithm supports update* INSERT(X)*, which appends an object X (as X_{t+1}) to the sequence, and query* ESTIMATE(s)*, which outputs a $(1 \pm \varepsilon)$-approximation to $\mathrm{vol}(X_s \cup \cdots \cup X_t)$ with high probability. The update time is $O(\log^2(n/\varepsilon)/\varepsilon^2)$, the query time is $O(\log(n)/\varepsilon^2)$, and the space complexity is $O(\log(M)\log(n)/\varepsilon^2)$.*

Our algorithm not only generalizes previous insertion-only results [40], but is also remarkably simple. Note that the space bound depends on the aspect ratio M, defined as the ratio of the largest volume of any object ever inserted and the smallest volume of any object ever inserted. In many applications, M is a polynomial in n and thus $\log(M) = O(\log n)$. Furthermore, geometric algorithms in the sliding window setting [24, 20] usually have logarithmic dependencies on parameters such as the aspect ratio or spread.

2.3 Streaming algorithm for convex bodies with insertions and deletions

Lastly, we consider streaming algorithms that support *both* insertions and deletions. We manage to develop an efficient algorithm for a reasonably wide class of geometric objects – namely, convex bodies in any constant dimension:

▶ **Theorem 3.** *Given $R \geq r > 0$, there is an algorithm that implicitly maintains a multiset $\mathcal{X}$ of convex bodies in $[0, R]^d$, where each convex body contains a ball of radius r, under updates* INSERT(X) *and* DELETE(X)*. Upon query* ESTIMATE$()$*, it outputs a $(1 \pm \varepsilon)$-approximation to $\mathrm{vol}\left(\bigcup_{X \in \mathcal{X}} X\right)$ with high probability. The expected update and query time and space are $O(\mathrm{polylog}(\frac{nR}{\varepsilon r})/\varepsilon^2)$ for constant d.*

The bounds depend polylogarithmically on R/r, which is common in volume estimation algorithms for a single convex object [23] as well as geometric streaming algorithms [29, 26].

Previously, Tirthapura and Woodruff obtained an efficient dynamic streaming algorithm only for axis-aligned boxes [47] (with $\mathrm{poly}(\log n, \log \Delta, \frac{1}{\varepsilon}, d)$ update time and space in a discrete universe $\{1, \ldots, \Delta\}^d$). To emphasize the generality of our algorithm, note that many non-convex objects can be decomposed into a small number of convex objects, e.g., non-convex polyhedra of constant combinatorial complexity in constant dimensions can be decomposed into a constant number of simplices, to which our algorithm can be applied.

3 Technical overview

3.1 Dynamic algorithm with insertions and deletions

Karp, Luby and Madras' algorithm [34] and subsequent work are based on the observation that we can estimate the volume of a union U by maintaining a finite random subset $S^{(\ell)} \subset U$ that selects every point independently with probability density $1/2^\ell$. It turns out that the "right" choice of ℓ is one such that $|S^{(\ell)}| \approx \log(n)/\varepsilon^2$; in this case $2^\ell |S^{(\ell)}|$ provides a good estimate of the volume of U.

Meel, Vinodchandran, and Chakraborty [40] described a particularly simple way to maintain such a random set $S^{(\ell)}$ for insertion-only updates: when a new object X is inserted, we remove from $S^{(\ell)}$ all points that are contained in X, then add to $S^{(\ell)}$ freshly sampled points in X with density $1/2^\ell$. This step is inexpensive for the right choice of ℓ since $|S^{(\ell)}|$ is logarithmic. If $|S^{(\ell)}|$ becomes too big, we increment ℓ (thereby halving the sampling rate and also $|S^{(\ell)}|$ in expectation) and move on.

Difficulty arises in the presence of deletions: the volume is no longer monotonically increasing, and the right choice of ℓ may increase or decrease over time. Our first observation is that the *deletion-only* case can be handled by a variant of Meel, Vinodchandran, and

Chakraborty's insertion-only algorithm. Roughly, we can maintain $S^{(\ell)}$ for the right ℓ by keeping a counter on the number of objects containing each sample point (recall that $|S^{(\ell)}|$ is logarithmic for the right ℓ). When an object X is deleted, we decrement the counters of all points in $S^{(\ell)} \cap X$, and when a counter drops to 0, the point is removed from $S^{(\ell)}$. When the number of surviving sample points drops below logarithmic, we decrease ℓ, rebuild $S^{(\ell)}$ by iterating over all objects that has not been deleted (recall that we can store pointers to all objects), and reinitialize the counters.

A popular technique in dynamic computational geometry by Bentley and Saxe [4], called the *logarithmic method*, transforms deletion-only data structures into fully dynamic ones. It works by decomposing a fully dynamic set into logarithmically many deletion-only subsets, increasing the update and query time by typically just a logarithmic factor. The logarithmic method applies to so-called *decomposable* problems that satisfy the following property: if the input is decomposed into subsets, then the answer to the original problem can be quickly obtained from the answers to the individual subsets. Unfortunately, union volume estimation is *not* a decomposable problem: We cannot determine the volume of the union of $\mathcal{X}_0 \cup \cdots \cup \mathcal{X}_{\log n}$ just from the volume of the union of each $\mathcal{X}_i$.

Nevertheless, we show that a novel variant of the logarithmic method can still work. Recall that the deletion-only data structure already maintains a sample for each $\mathcal{X}_i$. For each sample point we keep track of more counters – specifically, how many sets from the other $\mathcal{X}_i$'s contain the point. These counters provide enough information to avoid double counting and help us estimate the union volume accurately. One issue though is that many counters need to be adjusted when one of the deletion-only data structures changes ℓ and recomputes $S^{(\ell)}$. Fortunately, we manage to bound the total number of such changes, so the amortized cost for maintaining all counters remains polylogarithmic. See Section 5 for details.

3.2 Streaming algorithm with insertions and suffix queries

The dynamic algorithm with arbitrary insertions and deletions needs linear space because the data structure rebuilds on the remaining objects when the "right" ℓ decreases. However, in the sliding window setting and more generally the setting of suffix queries, the remaining objects are contiguous in time and need not be stored explicitly. We obtain a streaming algorithm in this case by a simple yet non-trivial generalization of Meel, Vinodchandran, and Chakraborty's algorithm. Specifically, we keep the timestamp when each point in $S^{(\ell)}$ is sampled. For each ℓ we keep track of the largest timestamp $s^{(\ell)}$ such that discarding points older than $s^{(\ell)}$ from $S^{(\ell)}$ makes $|S^{(\ell)}|$ logarithmically bounded. This timestamp $s^{(\ell)}$ is monotonically increasing over time for each fixed ℓ. To estimate the volume of the union of $X_s \cup \cdots \cup X_t$ for a given s at time t, we identify the smallest ℓ for which $s^{(\ell)} \le s$, and output $2^\ell |S^{(\ell)} \cap (X_s \cup \ldots \cup X_t)|$. See Section 6 in the full version [5] for details.

3.3 Streaming algorithm for convex bodies with insertions and deletions

Our streaming algorithm for low-dimensional convex objects for arbitrary insertions and deletions is based on another simple sampling strategy. First, it is not difficult to discretize the space $\mathbb{R}^d$ to a grid $[\Delta]^d$ for some large Δ, so that the volume of the union U is close to the number of grid points in $U \cap [\Delta]^d$. The idea now is to select a random subset $R^{(\ell)}$ of the grid $[\Delta]^d$ in the beginning, and maintain the intersection $S^{(\ell)} := U \cap R^{(\ell)}$ during the updates. Using a pairwise independent family of hash functions, $R^{(\ell)}$ can be represented implicitly with logarithmically many bits.

When we insert or delete an object X, we just need to insert or delete $X \cap R^{(\ell)}$ to or from $S^{(\ell)}$ (viewed as a multiset of points). An issue is that the space usage may be large for a $S^{(\ell)}$ with a "wrong" sampling rate, and we do not know which ℓ is the right choice beforehand. Furthermore, a wrong choice may become right at a later time. This issue can be resolved by representing $S^{(\ell)}$ by data structures for sparse vectors [26, 31], which support adding numbers to a given coordinate, as well as recovering the vector when its norm is below a threshold. In this way, we essentially just pay for the space at the right sampling rate.

A question remains: how to enumerate the points of $X \cap R^{(\ell)}$ for a given object X and for our implicitly represented set $R^{(\ell)}$? For the case where the objects are rotated (non-axis-aligned) boxes in constant dimension, and for a particular pairwise independent family of hash functions, we observe that this subproblem directly reduces to integer linear programming (ILP) with a constant number of constraints and variables, and therefore can be solved efficiently by well-known algorithms [38]. For more general convex objects in constant dimension, we bound each object X by a small rotated box $B \supseteq X$, use an ILP to enumerate the points of $B \cap R^{(\ell)}$ and discard those not in X. By "small" we mean that the volume of B is comparable to that of X, so in particular the number of discarded points is small. See Section 7 in the full version [5] for details.

Our approach is related to Tirthapura and Woodruff's algorithm [47] for axis-aligned boxes. Their idea was to directly adapt a streaming algorithm for approximately counting distinct elements to estimate $|U \cap [\Delta]^d|$: to insert or delete an object X, we cannot afford to literally insert or delete all points in $|X \cap [\Delta]^d|$, but they showed how to efficiently simulate the effect for a specific distinct elements algorithm in the case of axis-aligned boxes. Our approach is more modular in that we apply sparse recovery only as a black box. Our main contribution however is the realization, via solving an ILP, that this type of approach works far more generally than for axis-aligned boxes, namely, for general convex objects.

4 Preliminaries

We assume that all relevant parameters (the precision ε, an upper bound n on the number of operations, etc.) are given at initialization. Knowing n is in line with the usual assumptions in streaming algorithms; it can easily be avoided but this complicates notation. The phrase "with high probability" means that the probability is at least $1 - n^{-\Omega(1)}$. We denote $[n] := \{1, 2 \ldots, n\}$. The notation $a \in b \pm c$ means that $b - c \le a \le b + c$. Similarly, $a \in (1 \pm \varepsilon)b$ means that $(1 - \varepsilon)b \le a \le (1 + \varepsilon)b$, and we say that a is a $(1 \pm \varepsilon)$-approximation to b.

As in all previous work, our algorithms exploit the close relation between volume estimation and sampling. The notion of ℓ-samples is central:

▶ **Definition 4** (ℓ-sample). *Let U be a measurable set and $\ell \in \mathbb{Z}$. A random finite subset $S \subset U$ is an* ℓ-sample *of U, denoted $S \sim \mathrm{Pois}(U, \ell)$, if $|S \cap V| \sim \mathrm{Pois}(\mathrm{vol}(V)/2^\ell)$ for every measurable subset $V \subseteq U$.*

It follows from this definition that: (i) if $S \sim \mathrm{Pois}(U, \ell)$, then $S \cap V \sim \mathrm{Pois}(V, \ell)$ for any measurable subset $V \subseteq U$; (ii) if $S \sim \mathrm{Pois}(U, \ell)$ and $T \sim \mathrm{Pois}(V, \ell)$ are independent and U, V are disjoint, then $S \cup T \sim \mathrm{Pois}(U \cup V, \ell)$.

▶ **Definition 5** (level). *For a measurable set U, we define* $\mathrm{level}(U)$ *as the unique integer such that* $\frac{32 \log n}{\varepsilon^2} \le \frac{\mathrm{vol}(U)}{2^{\mathrm{level}(U)}} < \frac{64 \log n}{\varepsilon^2}$.

We will see later that if $S \sim \mathrm{Pois}(U, \ell)$ where $\ell \approx \mathrm{level}(U)$, then $|S|$ concentrates around its expectation $\mathrm{vol}(U)/2^\ell$.

Let U denote the union we are interested in. Each of our algorithms morally maintains an ℓ-sample $S^{(\ell)}$ of U for every $\ell \in \mathbb{Z}$ throughout the updates; upon query, it identifies some $L \approx \text{level}(U)$ and outputs $2^L|S^{(L)}|$. However, there are two pitfalls. First, the identification of L actually requires inspecting the outcomes of the random sets $S^{(\ell)}$, so L is a random variable and this complicates the concentration analysis. Second, we cannot afford to maintain all sets $S^{(\ell)}$ at once. Our algorithms only explicitly store those $S^{(\ell)}$ with $\ell \approx \text{level}(U)$; for such ℓ we have $\mathbb{E}|S^{(\ell)}| \approx \text{vol}(U)/2^{\text{level}(U)} = \Theta(\log(n)/\varepsilon^2)$. For $\ell \ll \text{level}(U)$ the set $S^{(\ell)}$ is too large and must be compressed or discarded. Naturally, this further complicates the algorithms.

5 Dynamic algorithm with insertions and deletions

In this section, we prove Theorem 1 by presenting a fully dynamic (non-streaming) algorithm for union volume estimation in linear space and polylogarithmic update time. Throughout we assume that the volume of each object is in the range $[(3n/\varepsilon)^2, (3n/\varepsilon)^4]$. The general case can be easily reduced to this case; see Lemma 4.1 in the full version [5].

5.1 Preparations

We will use the Karp–Luby–Madras algorithm as a subroutine:

▶ **Theorem 6** (Karp, Luby, Madras [34]). *Let $n \in \mathbb{N}$ be a parameter. There is an algorithm $\text{KLM}(\mathcal{X})$ that computes a (1 ± 0.5)-approximation to $\text{vol}(\bigcup_{X\in\mathcal{X}} X)$ with probability at least $1 - n^{-3}$ and runs in time $O(|\mathcal{X}| \log n)$.*

We need a somewhat technical notion that is "close" to an ℓ-sample at the right level ℓ.

▶ **Definition 7.** *Let U be a measurable set. A family of pairs $\{(\ell_1, R_1), \dots, (\ell_N, R_N)\}$ is called an* ensemble *of U if $\ell_k \in \text{level}(U) \pm 1$ and $R_k \sim \text{Pois}(U, \ell_k)$ for all $k \in [N]$. A random pair (L, S) is called a* (δ, N)-digest *of U if there exists an ensemble $\mathcal{E}$ of U such that $|\mathcal{E}| \le N$ and $\mathbb{P}[(L, S) \notin \mathcal{E}] \le \delta$. The ensemble $\mathcal{E}$ is called a* support ensemble *of the digest (L, S).*

▶ **Lemma 8.** *Fix measurable sets $V \subseteq U$. If $\mathcal{E}$ is an ensemble of U, then with probability at least $1 - \frac{2|\mathcal{E}|}{n^4}$, we have $\left|2^\ell\,|R \cap V| - \text{vol}(V)\right| \le \varepsilon\,\text{vol}(U)$ for all $(\ell, R) \in \mathcal{E}$.*

Proof. Consider an arbitrary $(\ell, R) \in \mathcal{E}$. By definition, $\ell \le \text{level}(U) + 1$ and $R \sim \text{Pois}(U, \ell)$. In particular, $|R \cap V| \sim \text{Pois}(\lambda)$ where $\lambda := \text{vol}(V)/2^\ell$. Let us write $d := \varepsilon\,\text{vol}(U)/2^\ell \ge \varepsilon\lambda$. By concentration of Poisson variables [11],

$$\begin{aligned}
&\mathbb{P}\left(\left|2^\ell\,|R \cap V| - \text{vol}(V)\right| > \varepsilon\,\text{vol}(U)\right) \;=\; \mathbb{P}(||R \cap V| - \lambda| > d) \\
&\le 2\exp\left(-\frac{d^2}{2(\lambda + d)}\right) \le 2\exp\left(-\frac{\varepsilon d}{2(1+\varepsilon)}\right) = 2\exp\left(-\frac{\varepsilon^2\,\text{vol}(U)}{2(1+\varepsilon)\cdot 2^\ell}\right) \le 2\mathrm{e}^{-4\log n} \le \frac{2}{n^4}
\end{aligned}$$

where the second last step uses $\frac{\text{vol}(U)}{2^\ell} \ge \frac{\text{vol}(U)}{2^{\text{level}(U)+1}} \ge \frac{16\log n}{\varepsilon^2}$. Taking a union bound over all $(\ell, R) \in \mathcal{E}$ proves the lemma. ◀

▶ **Corollary 9.** *Fix measurable sets $V \subseteq U$. If S is a (δ, N)-digest of U, then*

$$\mathbb{P}\left(\left|2^L\,|S \cap V| - \text{vol}(V)\right| \le \varepsilon\,\text{vol}(U)\right) \ge 1 - \delta - \frac{2N}{n^4}.$$

Proof. Let $\mathcal{E}$ be a support ensemble of (L, S), so $|\mathcal{E}| \le N$ and $\mathbb{P}[(L, S) \notin \mathcal{E}] \le \delta$. The event in the statement is implied by $(L, S) \in \mathcal{E}$ and the event in Lemma 8. ◀

▶ **Lemma 10.** *Let $n \in \mathbb{N}$ and $\varepsilon \in (0, \frac{1}{4}]$ be parameters. There is an algorithm* ERASE-RESAMPLE$(\mathcal{X}, L)$ *that, given a multiset $\mathcal{X}$ of $m \le n$ objects and a random variable L, returns a set S of size $|S| = O(\log(n)/\varepsilon^2)$ such that (L, S) is a $(\delta, 3)$-digest of $U := \bigcup_{X \in \mathcal{X}} X$ for $\delta := \frac{2}{n^3} + \mathbb{P}[L \notin \text{level}(U) \pm 1]$. The algorithm runs in time $O(m \log(n)/\varepsilon^2)$.*

Algorithm 1 Digesting a decremental union.

```
function ERASE-RESAMPLE(𝒳, L)
    S := ∅
    foreach X ∈ 𝒳 do
        foreach x ∈ S do
            if X.CONTAINS(x) then
                remove x from S
        sample a random number N ~ Pois(X.SIZE/2^L)
        if |S| + N > 160 log(n)/ε^2 then
            return ∅
        for k = 1, ..., N do
            x := X.SAMPLE()
            add x to S
    return S
```

Proof. The algorithm is described as Algorithm 1. The bound on the size $|S|$ is clear. The running time analysis is also easy: there are m iterations, each running in time $O(\log(n)/\varepsilon^2)$ due to the bound on $|S|$ (and on $|S| + N$).

Next, we show the correctness. Assume first that the algorithm does not return from the if-clause. Let $X_1, \dots, X_m$ be the objects in $\mathcal{X}$ in the order enumerated by the foreach-loop, and write $U := \bigcup_{i=1}^m X_i$. We define the random set $X_i^{(L)}$ as the set of points sampled from X_i in the for-k-loop; observe that $X_i^{(L)} \sim \text{Pois}(X_i, L)$ for every $i \in [n]$. Let S_i be the set S at the end of the iteration for $X = X_i$. Note that with $S_0 := \emptyset$ the algorithm computes $S_i = (S_{i-1} \setminus X_i) \cup X_i^{(L)}$ in each iteration. It follows that $S_i = \bigcup_{j=1}^i X_j^{(L)} \setminus (X_{j+1} \cup \dots \cup X_i)$. Since the random sets $X_j^{(L)} \setminus (X_{j+1} \cup \dots \cup X_i) \sim \text{Pois}(X_j \setminus (X_{j+1} \cup \dots \cup X_i), L)$ are L-samples with disjoint supports and the union of supports is $\bigcup_{j=1}^i X_j$, we have $S_i \sim \text{Pois}(\bigcup_{j=1}^i X_j, L)$.

Now let us analyze the probability of the algorithm aborting. Observe that the algorithm aborts if and only if $|S_i| > 160 \log(n)/\varepsilon^2$ for some $i \in [n]$. By the monotonicity of the size of the support $\bigcup_{j=1}^i X_j$, we have $\mathbb{P}[|S_1| > 160 \log(n)/\varepsilon^2] \le \dots \le \mathbb{P}[|S_n| > 160 \log(n)/\varepsilon^2]$. Fix $L \in \text{level}(U) \pm 1$. Since $(1+\varepsilon)\frac{\text{vol}(U)}{2^L} \le (1+\varepsilon)\frac{\text{vol}(U)}{2^{\text{level}(U)-1}} \le (1+\varepsilon)\frac{128 \log n}{\varepsilon^2} \le \frac{160 \log n}{\varepsilon^2}$, we have

$$\mathbb{P}\left(|S_n| > \frac{160 \log n}{\varepsilon^2}\right) \le \mathbb{P}\left(|S_n| > (1+\varepsilon)\frac{\text{vol}(U)}{2^L}\right) \le \frac{2}{n^4},$$

where the second inequality follows from Lemma 8 by interpreting (L, S_n) as a size-1 ensemble of U and plugging in $V := U$. By a union bound over all events $|S_i| > 160 \log(n)/\varepsilon^2$, it follows that the algorithm aborts with probability at most $\frac{2}{n^3}$, if $L \in \text{level}(U) \pm 1$.

If the algorithm does not abort, then we have shown above that it returns a set $S = S_n \sim \text{Pois}(\bigcup_{j=1}^n X_j, L) = \text{Pois}(U, L)$. It follows that we can compare the distribution of (L, S) with the size-3 ensemble $\mathcal{E} := \{(\ell, \text{Pois}(U, \ell)) \colon \ell \in \text{level}(U) \pm 1\}$. We can bound the error probability $\mathbb{P}[(L, S) \notin \mathcal{E}]$ by the probability that $L \notin \text{level}(U) \pm 1$ plus the probability that the algorithm aborts assuming $L \in \text{level}(U) \pm 1$, and we have bounded the latter probability by $\frac{2}{n^3}$. This proves the claim. ◀

5.2 Digesting decremental union

Next we study the decremental setting. That is, we receive a set of objects $\mathcal{X}$ and want to build a data structure that allows efficient deletions. At any time, it needs to quickly report a digest of the union of the surviving objects (i.e., objects that have not been deleted).

Algorithm 2 Digest data structure for decremental union.

parameters: n, ε
public variables: $\mathcal{X}, S, L$
function INITIALIZE($\mathcal{Y}$)
  let $\mathcal{X} := \mathcal{Y}$ and $L := \infty$
  REFRESH()
function REFRESH
  $v := \mathrm{KLM}(\mathcal{X})$
  let L' be the unique integer such that $\frac{32 \log n}{\varepsilon^2} \le v/2^{L'} < \frac{64 \log n}{\varepsilon^2}$
  $L := \min\{L - 1, L'\}$
  $S :=$ ERASE-RESAMPLE($\mathcal{X}, L$)
  foreach $x \in S$ **do**
    compute and store $m_x := \#\{X \in \mathcal{X} : X.\text{CONTAINS}(x)\}$
function DELETE(X)
  $\mathcal{X} := \mathcal{X} \setminus X$
  foreach $x \in S$ **do**
    if $X.\text{CONTAINS}(x)$ **then**
      $m_x := m_x - 1$
    if $m_x = 0$ **then**
      $S := S \setminus \{x\}$ and forget m_x
  if $|S| < \frac{24 \log n}{\varepsilon^2}$ **then**
    REFRESH()
    return true
  else
    return false

▶ **Lemma 11.** *Let $n \in \mathbb{N}$ and $\varepsilon \in (0, \frac{1}{16}]$ be parameters. Algorithm 2 supports* INITIALIZE($\mathcal{X}$)*, which initializes with a multiset $\mathcal{X}$ of $m \le n$ objects whose volumes are in $[(3n/\varepsilon)^2, (3n/\varepsilon)^4]$, and* DELETE($X$)*, which removes object X from $\mathcal{X}$. The algorithm maintains a pair (L, S) which is a $(\frac{18}{n^2}, 6n)$-digest of $\bigcup_{X \in \mathcal{X}} X$ at any point in time, and $|S| = O(\log(n)/\varepsilon^2)$. In total,* INITIALIZE *and any sequence of* DELETE *operations run in expected time $O(m \log^2(n/\varepsilon)/\varepsilon^2)$. The space complexity is $O(m + \log(n)/\varepsilon^2)$.*

Moreover, the set S only rarely gains new points: DELETE(X) *returns true if the set S after the deletion is not a subset of the set S before the deletion (and false otherwise); and over any sequence of deletions only $O(\log(n/\varepsilon))$ deletions return true, with probability $1 - O(n^{-2})$.*

Proof. Fix the initial objects and an arbitrary sequence of m deletions. After $t = 0, 1, \ldots, m$ deletions, let U_t be the union of the surviving objects, and let L_t, S_t be the variables L, S in the algorithm. We claim that (L_t, S_t) is a $(\frac{9(t+1)}{n^3}, 3(t+1))$-digest of U_t.

Let us prove the claim by induction. First consider initialization ($t = 0$). By Theorem 6, with probability at least $1 - n^{-3}$ KLM computes $v \in (1 \pm 0.5)\,\mathrm{vol}(U_0)$, and in this case the algorithm computes $L \in \mathrm{level}(U_0) \pm 1$. Thus, ERASE-RESAMPLE returns a $(\frac{3}{n^3}, 3)$-digest (L_0, S_0) of U_0 by Lemma 10.

Now we proceed to $t \geq 1$. By the induction hypothesis, (L_{t-1}, S_{t-1}) is a $(\frac{9t}{n^3}, 3t)$-digest of U_{t-1}. Let $\mathcal{E}$ be a support ensemble of (L_{t-1}, S_{t-1}). Define

$$\mathcal{E}' := \{(\ell, R \cap U_t) : (\ell, R) \in \mathcal{E},\ \ell \leq \text{level}(U_t) + 1\}.$$

Observe that $|\mathcal{E}'| \leq |\mathcal{E}| \leq 3t$ and $\mathcal{E}'$ is an ensemble of U_t since $\text{level}(U_t) \leq \text{level}(U_{t+1})$.

Let E be the event that $(L_{t-1}, S_{t-1}) \in \mathcal{E}$. Let F be the event that $\left|2^\ell |R \cap U_t| - \text{vol}(U_t)\right| \leq \varepsilon\,\text{vol}(U_{t-1})$ for all $(\ell, R) \in \mathcal{E}$. Note that $\mathbb{P}(\overline{E}) \leq \frac{9t}{n^3}$ by the induction hypothesis, and $\mathbb{P}(\overline{F}) \leq \frac{2 \cdot 3t}{n^4} < \frac{6}{n^3}$ by Lemma 8.

Assuming $E \cap F$, we have $L_{t-1} \in \text{level}(U_{t-1}) \pm 1$ and $2^{L_{t-1}} |S_{t-1} \cap U_t| \in \text{vol}(U_t) \pm \varepsilon\,\text{vol}(U_{t-1})$. Let us distinguish three cases and derive further consequences:

(1) If $L_{t-1} \leq \text{level}(U_t)$, then

$$|S_{t-1} \cap U_t| \geq \frac{\text{vol}(U_t)}{2^{\text{level}(U_t)}} - \varepsilon \frac{\text{vol}(U_{t-1})}{2^{\text{level}(U_{t-1})-1}} \geq \frac{32 \log n}{\varepsilon^2} - \varepsilon \frac{128 \log n}{\varepsilon^2} \geq \frac{24 \log n}{\varepsilon^2}.$$

So the algorithm does not refresh. As a result, $L_t = L_{t-1} \leq \text{level}(U_t)$ and $S_t = S_{t-1} \cap U_t$, thus $(L_t, S_t) \in \mathcal{E}'$.

(2) If $L_{t-1} \geq \text{level}(U_t) + 2$, then

$$|S_{t-1} \cap U_t| \leq \frac{\text{vol}(U_t)}{2^{\text{level}(U_t)+2}} + \varepsilon \frac{\text{vol}(U_{t-1})}{2^{\text{level}(U_{t-1})-1}} < \frac{16 \log n}{\varepsilon^2} + \varepsilon \frac{128 \log n}{\varepsilon^2} \leq \frac{24 \log n}{\varepsilon^2}.$$

So the algorithm refreshes. During refresh, the algorithm computes $L' \in \text{level}(U_t) \pm 1$ with probability at least $1 - \frac{1}{n^3}$ by Theorem 6, and then we have $L_t = \min\{L_{t-1} - 1, L'\} = L' \in \text{level}(U_t) \pm 1$. Hence, it returns a $(\frac{3}{n^3}, 3)$-digest (L_t, S_t) of U_t by Lemma 10.

(3) Otherwise $L_{t-1} = \text{level}(U_t)+1$, and the algorithm may or may not refresh. If it does, then (L_t, S_t) is a $(\frac{3}{n^3}, 3)$-digest of U_t as in case (2). If it does not, then $L_t = L_{t-1} = \text{level}(U_t)+1$ and $S_t = S_{t-1} \cap U_t$, thus $(L_t, S_t) \in \mathcal{E}'$.

Taking all cases into account, recalling $\mathbb{P}(\overline{E \cap F}) \leq \mathbb{P}(\overline{E}) + \mathbb{P}(\overline{F}) \leq \frac{9t+6}{n^3}$, and adding the error probability $\frac{3}{n^3}$ of the new digest after refreshing, we conclude that (L_t, S_t) is a $(\frac{9(t+1)}{n^3}, 3(t+1))$-digest of U_t. (In particular, it is a $(\frac{18}{n^2}, 6n)$-digest of the union of the remaining objects.)

As a consequence, with probability at least $1 - O(n^{-2})$ we have $L_0 \in \text{level}(U_0) \pm 1$ and $L_{m-1} \in \text{level}(U_{m-1}) \pm 1$. On the other hand, $\text{level}(U_0) \leq O(\log(n/\varepsilon))$ and $\text{level}(U_{m-1}) \geq 0$ (because each object has volume in $[(3n/\varepsilon)^2, (3n/\varepsilon)^4]$), and every refresh causes L to decrease by at least one. So the number of calls to REFRESH is bounded by $O(\log(n/\varepsilon))$. Since we have $S_t \subseteq S_{t-1}$ except when we do a refresh, it follows that the set S can only gain points during $O(\log(n/\varepsilon))$ many deletions, with probability $1 - O(n^{-2})$.

The size bound $|S| = O(\log(n)/\varepsilon^2)$ follows from Lemma 10.

It remains to bound the running time and space usage. For INITIALIZE and REFRESH, the subroutine KLM runs in $O(m \log n)$ time by Theorem 6; the subroutine ERASE-RESAMPLE runs in $O(m \log(n)/\varepsilon^2)$ time by Lemma 10; and the loop runs in $O(m \log(n)/\varepsilon^2)$ time since $|S| = O(\log(n)/\varepsilon^2)$ and each point in S is tested against at most m objects.

For DELETE, we spend $O(\log(n)/\varepsilon^2)$ time in the loop. An extra $O(m \log(n)/\varepsilon^2)$ time may be incurred by a refresh. As we argued before, there are at most $O(\log(n/\varepsilon))$ refreshes throughout the deletion sequence with probability $1 - O(n^{-2})$. There can be at most m refreshes with the remaining probability $O(n^{-2})$. So the expected time for the whole deletion sequence is $O(m \log(n)/\varepsilon^2 \cdot (\log(n/\varepsilon) + m/n^2)) \leq O(m \log^2(n/\varepsilon)/\varepsilon^2)$.

Finally, let us analyze the space complexity. Storing $\mathcal{X}$ in a binary search tree needs $O(m)$ space. Storing S and the counters m_x in a table needs $O(\log(n)/\varepsilon^2)$ space. Storing L needs constant space. So the algorithm uses $O(m + \log(n)/\varepsilon^2)$ space in total. ◀

5.3 Handling insertions

In this section, we apply the decremental data structure from the previous section to derive a dynamic algorithm for union volume estimation, using a novel variant of the logarithmic method [4]. Throughout the algorithm, the remaining objects are partitioned into bins $\mathcal{X}_0, \dots, \mathcal{X}_{\log n}$. Each bin $\mathcal{X}_j$ holds at most 2^j objects and is implemented by the decremental data structure. Every now and then, we merge some (small) bins into a larger bin by re-initializing the target bin with the objects in the source bins. Between any two merges that involve a bin $\mathcal{X}_j$, it undergoes deletions only. At any time, we can access a digest $(\mathcal{X}_j.L, \mathcal{X}_j.S)$ of the union $\bigcup_{X \in \mathcal{X}_j} X$. Additionally, for every point $x \in \mathcal{X}_j.S$, we keep track of the number $n_x := \#\{X \in \mathcal{X}_0 \cup \dots \cup \mathcal{X}_{j-1} : x \in X\}$.

▶ **Theorem 12.** *Let $n \in \mathbb{N}$ and $\hat{\varepsilon} \in (0, \frac{1}{16}]$ be parameters. Algorithm 3 maintains an initially empty multiset $\mathcal{X}$ of objects, where each object has volume in $[(3n/\hat{\varepsilon})^2, (3n/\hat{\varepsilon})^4]$, under updates* INSERT$(X)$ *and* DELETE(X) *and query* ESTIMATE$()$*, which $(1 \pm \hat{\varepsilon})$-approximates the volume of the union of $\mathcal{X}$ and is correct with high probability. The amortized expected update time is $O(\log^5(n/\hat{\varepsilon})/\hat{\varepsilon}^2)$, the query time is $O(\log^4(n)/\hat{\varepsilon}^2)$, and the space complexity is $O(n + \log^4(n)/\hat{\varepsilon}^2)$.*

Algorithm 3 Dynamic union volume estimation with arbitrary insertions and deletions.

parameters: $n, \hat{\varepsilon}$
function INITIALIZE
  for $j = 0, \dots, \log n$ **do**
    $\mathcal{X}_j$.INITIALIZE$(\emptyset)$ with parameters n and $\varepsilon := \hat{\varepsilon}/(1 + \log n)$
function INSERT(X)
  let h be the minimal integer such that $1 + \sum_{j=0}^{h} |\mathcal{X}_j| \le 2^h$
  $\mathcal{X}_h$.INITIALIZE$(\{X\} \cup \mathcal{X}_0 \cup \dots \cup \mathcal{X}_h)$ `/* merge */`
  $\mathcal{X}_j$.INITIALIZE$(\emptyset)$ for $j = 0, \dots h-1$
  foreach $x \in \mathcal{X}_h.S$ **do**
    $n_x := 0$
  foreach $x \in \bigcup_{j=h+1}^{\log n} \mathcal{X}_j.S$ **do**
    if X.CONTAINS(x) **then**
      $n_x := n_x + 1$
function DELETE(X)
  find the bin $\mathcal{X}_j$ containing X
  refreshed := $\mathcal{X}_j$.DELETE(X)
  if refreshed **then**
    foreach $x \in \mathcal{X}_j.S$ **do**
      $n_x := \#\{X \in \mathcal{X}_0 \cup \dots \cup \mathcal{X}_{j-1} : X$.CONTAINS$(x)\}$
  foreach $x \in \bigcup_{k=j}^{\log n} \mathcal{X}_k.S$ **do**
    if X.CONTAINS(x) **then**
      $n_x := n_x - 1$
function ESTIMATE$()$
  denote $S_j := \mathcal{X}_j.S$ and $L_j := \mathcal{X}_j.L$ for $j = 0, 1, \dots, \log n$
  compute $R_j := \{x \in S_j : n_x = 0\}$ for $j = 0, 1, \dots, \log n$
  return $\sum_{j=0}^{\log n} 2^{L_j} |R_j|$

Proof. Fix a sequence of at most n insertions and deletions. Then the partition of surviving objects into bins $\mathcal{X}_1, \dots, \mathcal{X}_{\log n}$ is determined; in particular $U_j := \bigcup_{X \in \mathcal{X}_j} X$ is determined, and we write $V_j := U_j \setminus (U_0 \cup \cdots \cup U_{j-1})$. Note that the union of the surviving objects $U := \bigcup_{j=0}^{\log n} U_j = \bigcup_{j=0}^{\log n} V_j$ is a disjoint union of V_j's, thus $\operatorname{vol}(U) = \sum_{j=0}^{\log n} \operatorname{vol}(V_j)$.

Now let us consider a call to ESTIMATE. By Lemma 11, each S_j is a $(\frac{18}{n^2}, 6n)$-digest of U_j. Note that the algorithm computes $R_j := V_j \cap S_j$, so by Corollary 9, we have $2^{L_j}|R_j| \in \operatorname{vol}(V_j) \pm \varepsilon \operatorname{vol}(U_j)$ with probability at least $1 - \frac{18}{n^2} - \frac{12n}{n^4} = 1 - O(n^{-2})$. By a union bound, this holds for all j simultaneously with probability $1 - O(n^{-1})$. Under this event,

$$\sum_{j=0}^{\log n} 2^{L_j}|R_j| \in \operatorname{vol}(U) \pm \varepsilon \sum_{j=0}^{\log n} \operatorname{vol}(U_j).$$

We can crudely bound the additive error by $\varepsilon(1 + \log n)\operatorname{vol}(U) = \hat{\varepsilon}\operatorname{vol}(U)$, just as desired.

Next we analyze the running time. We will charge the cost of DELETE to insertions. Note that in DELETE, it takes time $O(\log^2(n))$ to compute the bin j containing X, if for each j we store (pointers to) all objects in $\mathcal{X}_j$ in a binary search tree. The last for-loop takes time $O(\log^2(n)/\varepsilon^2)$. To bound the running time of the "if refreshed" block, we change our viewpoint and focus on the sequence of deletions occurring between two merges of a particular bin $\mathcal{X}_j$.

- The total cost of $\mathcal{X}_j$.DELETE is $O(2^j \log^2(n/\varepsilon)/\varepsilon^2)$ in expectation by Lemma 11.
- With probability $1 - O(n^{-2})$, there are $O(\log(n/\varepsilon))$ refreshes between two merges of bin $\mathcal{X}_j$ by Lemma 11, so the if-block is executed $O(\log(n/\varepsilon))$ times. Each time, the algorithm recomputes n_x for every $x \in S_j$ by testing against at most $\sum_{k=0}^{j-1} 2^k \le 2^j$ objects, which takes time $O(2^j \log(n)/\varepsilon^2)$. Therefore, the total contribution of the if-block is $O(2^j \log^2(n/\varepsilon)/\varepsilon^2)$ in expectation.

Therefore, over the sequence of deletions between two merges of $\mathcal{X}_j$, deletions from $\mathcal{X}_j$ take expected total time $O(2^j \log^2(n/\varepsilon)/\varepsilon^2)$. Note that at least 2^j insertions occur between two merges of bin $\mathcal{X}_j$. We can thus charge the cost of deletions to insertions. Each insertion is charged $O(\log n)$ times (once for every bin), so we charge each insertion with a cost of $O(\log^3(n/\varepsilon)/\varepsilon^2)$ in expectation.

Now we analyze the running time of INSERT. It depends on the value h. By Lemma 11, the merge step costs $O\left(\sum_{j=0}^{h} 2^j \log^2(n/\varepsilon)/\varepsilon^2\right) = O\left(2^h \log^2(n/\varepsilon)/\varepsilon^2\right)$, which can be expensive in the worst case. However, bin $\mathcal{X}_h$ is merged at most once every 2^h updates, so it contributes amortized cost $O(\log^2(n/\varepsilon)/\varepsilon^2)$ to each insertion. Summing over all h, the amortized cost of merges is $O(\log^3(n/\varepsilon)/\varepsilon^2)$ per insertion. Next we consider the foreach-loops. Since each digest contains $O(\log(n)/\varepsilon^2)$ points by Lemma 11, and the algorithm spends constant time on each point, these loops take time $O(\log^2(n)/\varepsilon^2)$. To summarize, INSERT runs in amortized expected time $O(\log^3(n/\varepsilon)/\varepsilon^2) = O(\log^5(n/\varepsilon)/\hat{\varepsilon}^2)$ (and DELETE runs in amortized time 0).

The running time of ESTIMATE is clearly $O(\log^2(n)/\varepsilon^2) = O(\log^4(n)/\hat{\varepsilon}^2)$.

The space of bin $\mathcal{X}_j$ is $O(2^j + \log(n)/\varepsilon^2)$ by Lemma 11, and we can charge the space of counters n_x to it. In total the space complexity is $O\left(\sum_{j=0}^{\log n}(2^j + \log(n)/\varepsilon^2)\right) = O(n + \log^4(n)/\hat{\varepsilon}^2)$. ◀

Theorem 1 now follows from Theorem 12 together with Lemma 4.1 in [5].

References

1 Pankaj K. Agarwal, Lars Arge, and Frank Staals. Improved dynamic geodesic nearest neighbor searching in a simple polygon. In *Proc. 34th International Symposium on Computational Geometry (SoCG)*, volume 99 of *LIPIcs*, pages 4:1–4:14, 2018. doi:10.4230/LIPIcs.SOCG.2018.4.

2 Pankaj K. Agarwal, Hsien-Chih Chang, Subhash Suri, Allen Xiao, and Jie Xue. Dynamic geometric set cover and hitting set. *ACM Transactions on Algorithms*, 18(4):40:1–40:37, 2022. doi:10.1145/3551639.

3 Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proc. 21st ACM Symposium on Principles of Database Systems (PODS)*, pages 1–16, 2002. doi:10.1145/543613.543615.

4 Jon Louis Bentley and James B Saxe. Decomposable searching problems I. Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980. doi:10.1016/0196-6774(80)90015-2.

5 Sujoy Bhore, Karl Bringmann, Timothy M. Chan, and Yanheng Wang. Dynamic and streaming algorithms for union volume estimation, 2026. arXiv:2602.16306.

6 Sujoy Bhore and Timothy M. Chan. Fast static and dynamic approximation algorithms for geometric optimization problems: Piercing, independent set, vertex cover, and matching. In *Proc. 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2357–2386, 2025. doi:10.1137/1.9781611978322.79.

7 Sujoy Bhore, Martin Nöllenburg, Csaba D. Tóth, and Jules Wulms. Fully dynamic maximum independent sets of disks in polylogarithmic update time. In *40th International Symposium on Computational Geometry (SoCG)*, volume 293 of *LIPIcs*, pages 19:1–19:16, 2024. doi:10.4230/LIPIcs.SOCG.2024.19.

8 Vladimir Braverman. Sliding window algorithms. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 2006–2011. Springer New York, New York, NY, 2016. doi:10.1007/978-1-4939-2864-4_797.

9 Karl Bringmann and Tobias Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Computational Geometry*, 43(6-7):601–610, 2010. doi:10.1016/J.COMGEO.2010.03.004.

10 Karl Bringmann, Kasper Green Larsen, André Nusser, Eva Rotenberg, and Yanheng Wang. Approximating Klee's measure problem and a lower bound for union volume estimation. In *Proc. 41st International Symposium on Computational Geometry (SoCG)*, pages 25:1–25:16, 2025. doi:10.4230/LIPIcs.SOCG.2025.25.

11 Clément Canonne. A short note on Poisson tail bounds, 2019. URL: https://github.com/ccanonne/probabilitydistributiontoolbox/blob/master/poissonconcentration.pdf.

12 Nofar Carmeli, Nikolaos Tziavelis, Wolfgang Gatterbauer, Benny Kimelfeld, and Mirek Riedewald. Tractable orders for direct access to ranked answers of conjunctive queries. *ACM Transactions on Database Systems*, 48(1):1–45, 2023. doi:10.1145/3578517.

13 Nofar Carmeli, Shai Zeevi, Christoph Berkholz, Benny Kimelfeld, and Nicole Schweikardt. Answering (unions of) conjunctive queries using random access and random-order enumeration. In *Proc. 39th ACM Symposium on Principles of Database Systems (PODS)*, pages 393–409, 2020. doi:10.1145/3375395.3387662.

14 Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. A scalable approximate model counter. In *Proc. 19th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 200–216, 2013. doi:10.1007/978-3-642-40627-0_18.

15 Timothy M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Computational Geometry*, 43(3):243–250, 2010. doi:10.1016/J.COMGEO.2009.01.007.

16 Timothy M. Chan. Klee's measure problem made easy. In *Proc. 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 410–419, 2013. doi:10.1109/FOCS.2013.51.

17 Timothy M. Chan, Qizheng He, Subhash Suri, and Jie Xue. Dynamic geometric set cover, revisited. *SIAM Journal on Computing*, 54(3):664–701, 2025. doi:10.1137/23M1596582.

18 Timothy M. Chan and Zhengcheng Huang. Dynamic geometric connectivity in the plane with constant query time. In *40th International Symposium on Computational Geometry (SoCG)*, volume 293 of *LIPIcs*, pages 36:1–36:13, 2024. doi:10.4230/LIPIcs.SOCG.2024.36.

19 Timothy M. Chan, Mihai Pătraşcu, and Liam Roditty. Dynamic connectivity: Connecting to networks and geometry. *SIAM Journal on Computing*, 40(2):333–349, 2011. doi:10.1137/090751670.

20 Timothy M. Chan and Bashir S. Sadjad. Geometric optimization problems over sliding windows. *Int. J. Comput. Geom. Appl.*, 16(2-3):145–158, 2006. doi:10.1142/S0218195906001975.

21 Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007. doi:10.1007/S00778-006-0004-3.

22 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813, 2002. doi:10.1137/S0097539701398363.

23 Martin E. Dyer, Alan M. Frieze, and Ravi Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. *Journal of the ACM*, 38(1):1–17, 1991. doi:10.1145/102782.102783.

24 Joan Feigenbaum, Sampath Kannan, and Jian Zhang. Computing diameter in the streaming and sliding-window models. *Algorithmica*, 41(1):25–41, 2005. doi:10.1007/S00453-004-1105-2.

25 Shiyuan Feng, William Swartworth, and David P. Woodruff. Tight bounds for heavy-hitters and moment estimation in the sliding window model. In *Proc. 52nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 75:1–75:19, 2025. doi:10.4230/LIPIcs.ICALP.2025.75.

26 Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. In *Proc. 21st ACM Symposium on Computational Geometry (SoCG)*, pages 142–149, 2005. doi:10.1145/1064092.1064116.

27 Egor Gorbachev and Marvin Künnemann. Combinatorial designs meet hypercliques: Higher lower bounds for Klee's measure problem and related problems in dimensions $d \geq 4$. In *Proc. 39th International Symposium on Computational Geometry (SoCG)*, pages 36:1–36:14, 2023. doi:10.4230/LIPIcs.SOCG.2023.36.

28 Monika Henzinger, Stefan Neumann, and Andreas Wiese. Dynamic approximate maximum independent set of intervals, hypercubes and hyperrectangles. In *36th International Symposium on Computational Geometry (SoCG)*, volume 164 of *LIPIcs*, pages 51:1–51:14, 2020. doi:10.4230/LIPIcs.SOCG.2020.51.

29 Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In *Proc. 36th annual ACM Symposium on Theory of Computing*, pages 373–380, 2004. doi:10.1145/1007352.1007413.

30 Rajesh Jayaram, David P. Woodruff, and Samson Zhou. Truly perfect samplers for data streams and sliding windows. In *Proc. 41st ACM Symposium on Principles of Database Systems (PODS)*, pages 29–40, 2022. doi:10.1145/3517804.3524139.

31 Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for L_p samplers, finding duplicates in streams, and related problems. In *Proc. 30th ACM Symposium on Principles of Database Systems (PODS)*, pages 49–58, 2011. doi:10.1145/1989284.1989289.

32 David R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM Review*, 43(3):499–522, 2001. doi:10.1137/S0036144501387141.

33 Richard M. Karp and Michael Luby. Monte-Carlo algorithms for the planar multiterminal network reliability problem. *Journal of Complexity*, 1(1):45–64, 1985. doi:10.1016/0885-064X(85)90021-4.

34 Richard M. Karp, Michael Luby, and Neal Madras. Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10(3):429–448, 1989. doi:10.1016/0196-6774(89)90038-2.

35 Benny Kimelfeld, Yuri Kosharovsky, and Yehoshua Sagiv. Query efficiency in probabilistic XML models. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 701–714, 2008. doi:10.1145/1376616.1376687.

36 Victor Klee. Can the measure of $\bigcup_1^n [a_i, b_i]$ be computed in less than $O(n \log n)$ steps? *The American Mathematical Monthly*, 84(4):284–285, 1977. doi:10.1080/00029890.1977.11994336.

37 Marvin Künnemann. A tight (non-combinatorial) conditional lower bound for Klee's measure problem in 3D. In *Proc. 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 555–566, 2022. doi:10.1109/FOCS54457.2022.00059.

38 Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. doi:10.1287/moor.8.4.538.

39 Kuldeep S. Meel, Sourav Chakraborty, and N. V. Vinodchandran. Estimation of the size of union of Delphic sets: Achieving independence from stream size. In *Proc. 41st ACM Symposium on Principles of Database Systems (PODS)*, pages 41–52, 2022. doi:10.1145/3517804.3526222.

40 Kuldeep S. Meel, N. V. Vinodchandran, and Sourav Chakraborty. Estimating the size of union of sets in streaming models. In *Proc. 40th ACM Symposium on Principles of Database Systems (PODS)*, pages 126–137, 2021. doi:10.1145/3452021.3458333.

41 Shanmugavelayutham Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2):117–236, 2005. doi:10.1561/0400000002.

42 Mridul Nandi, N. V. Vinodchandran, Arijit Ghosh, Kuldeep S. Meel, Soumit Pal, and Sourav Chakraborty. Improved streaming algorithm for the Klee's measure problem and generalizations. In *Proc. 27th International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX/RANDOM)*, pages 26:1–26:21, 2024. doi:10.4230/LIPIcs.APPROX/RANDOM.2024.26.

43 Yakov Nekrich. Dynamic planar point location in optimal time. In *53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1003–1014, 2021. doi:10.1145/3406325.3451100.

44 Mark H. Overmars and Chee-Keng Yap. New upper bounds in Klee's measure problem. *SIAM Journal on Computing*, 20(6):1034–1045, 1991. doi:10.1137/0220065.

45 A. Pavan, N. Variyam Vinodchandran, Arnab Bhattacharyya, and Kuldeep S. Meel. Model counting meets F_0 estimation. *ACM Transactions on Database Systems*, 48(3):1–28, 2023. doi:10.1145/3603496.

46 Christopher Ré, Nilesh Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *Proc. 23rd IEEE International Conference on Data Engineering*, pages 886–895, 2006. doi:10.1109/ICDE.2007.367934.

47 Srikanta Tirthapura and David Woodruff. Rectangle-efficient aggregation in spatial data streams. In *Proc. 31st ACM Symposium on Principles of Database Systems (PODS)*, pages 283–294, 2012. doi:10.1145/2213556.2213595.

48 David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1183–1196, 2021. doi:10.1109/FOCS52979.2021.00116.

Dynamic Light Spanners in Doubling Metrics

Sujoy Bhore
Department of Computer Science & Engineering, Indian Institute of Technology Bombay, India

Jonathan Conroy
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Arnold Filtser
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Abstract

A t-spanner of a point set X in a metric space $(\mathcal{X}, \delta)$ is a graph G with vertex set P such that, for any pair of points $u, v \in X$, the distance between u and v in G is at most t times $\delta(u, v)$. We study the problem of maintaining a spanner for a dynamic point set X – that is, when X undergoes a sequence of insertions and deletions – in a metric space of constant doubling dimension. For any constant $\varepsilon > 0$, we maintain a $(1+\varepsilon)$-spanner of P whose total weight remains within a constant factor of the weight of the minimum spanning tree of X. Each update (insertion or deletion) can be performed in $\text{poly}(\log \Phi)$ time, where Φ denotes the aspect ratio of X. Prior to our work, no efficient dynamic algorithm for maintaining a light-weight spanner was known even for point sets in low-dimensional Euclidean space.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Routing and network design problems

Keywords and phrases Dynamic data structures, spanners, light-weight, Euclidean metrics, doubling metrics

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.13

Related Version *Full Version*: https://arxiv.org/abs/2603.23490

Funding *Sujoy Bhore*: Work supported in part by ANRF ARG-MATRICS, Grant 002465.
Jonathan Conroy: Supported in part by U.S. National Science Foundation Grant No. CCF-2443017.
Arnold Filtser: This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 1042/22).

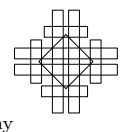
1 Introduction

Let $X \subset \mathcal{X}$ be a set of points in the metric $(\mathcal{X}, \delta_X)$. A graph $G = (X, E, \delta)$ with vertex set X is a *geometric graph* if the weight function on the edges is the distance function δ. In other words, the weight of each edge is the distance between its endpoints in $\mathcal{X}$. Observe that if G is a geometric graph, then by the triangle inequality, for every $x, y \in X$, $\delta_G(x, y) \geq \delta(x, y)$. Given a parameter t, called the *stretch*, we say that G is a (geometric) *t-spanner* for X if $\delta_G(x, y) \leq t \cdot \delta(x, y)$ for every $x, y \in X$. The natural goal is to construct, for a given stretch t, a t-spanner with small number of edges and a small total weight.

Lightness and *sparsity* are two central measures for evaluating spanners. For a spanner $G = (X, E)$, the lightness is the ratio $w(G)/w(\text{MST})$ between the total weight of G and the weight of a minimum spanning tree on X. The sparsity of G is the ratio $|E(G)|/|E(\text{MST})| \approx |E(G)|/|X|$, comparing the number of edges of G to that of an MST. Since every spanner (with finite stretch) is connected and therefore contains a spanning tree, the lightness and sparsity of G constitute a measure of how close a spanner (specifically $w(G)$ and $|E(G)|$) to the optimal weight and the optimal number of edges of a spanner with finite stretch. A long line of research has studied spanners in Euclidean spaces [5, 49, 41, 29, 32, 14, 45],



© Sujoy Bhore, Jonathan Conroy, and Arnold Filtser;
licensed under Creative Commons License CC-BY 4.0



doubling [35, 21, 51, 36, 33, 19, 13], planar and minor-free metric [18, 39, 25, 12], and general graphs [48, 4, 30, 28, 24, 33, 44, 16]. Of particular interest, we focus on *low-dimensional Euclidean space*, and more generally on *doubling metrics*.

Doubling metrics. Let $B(x,r) = \{y|\delta(x,y) \leq r\}$ be a ball of radius $r \geq 0$ centered at a point $x \in \mathcal{X}$. A metric $(\mathcal{X}, \delta)$ has *doubling dimension* d if for every $r > 0$, every ball of radius r can be covered by at most 2^d balls of radius $r/2$: $\forall x \in X, r > 0,\ \exists Y \subseteq X : |Y| \leq 2^d$ and $B_X(x,r) \subseteq \cup_{y \in Y} B_X(y, r/2)$. The study of spanners in doubling metrics was initiated by Gao, Guibas, and Nguyen [35], who proved that any n-point set in a metric space of doubling dimension d admits a $(1+\varepsilon)$-spanner with $\varepsilon^{-O(d)}n$ edges. Smid [51], via an analysis of the *greedy algorithm*, showed that greedy $(1+\varepsilon)$-spanners have $\varepsilon^{-O(d)}n$ edges and achieve lightness $\varepsilon^{-O(d)} \log n$. While it has long been known that d-dimensional Euclidean point sets admit $(1+\varepsilon)$-spanners with lightness $\varepsilon^{-O(d)}$, it remained a major open problem whether an analogous result holds for metric spaces of low doubling dimension. In a major breakthrough, Gottlieb [36] proved that any metric space of doubling dimension d admits a $(1+\varepsilon)$-spanner with lightness $(d/\varepsilon)^{O(d)}$. Gottlieb devised a complicated spanner construction instead of the classic greedy spanner. Later, Filtser and Solomon [33] showed that the greedy spanner is existentially optimal, concluding that it has lightness $(d/\varepsilon)^{O(d)}$ as well. Finally, Borradaile, Le, and Wulff-Nilsen [19] showed the greedy spanner has weight $\varepsilon^{-O(d)}$, which is optimal.

Dynamic Spanners. The problem of maintaining a sparse $(1+\varepsilon)$-spanner for a point set X under insertions and deletions has also been studied over the years. Already back in 1999, Arya et al. [6] obtained polylogarithmic update time in a restricted model with random updates (a deletion removes a uniformly random point from S, and an insertion results in a uniformly random point in the updated set). Later, Bose et al. [20] studied the problem in the context of θ-graphs, and presented an algorithm that supports insertions only and achieves polylogarithmic update time. Gao et al. [35] studied kinetic and dynamic geometric spanners for points in bounded doubling dimensions, and obtained a fully dynamic spanner with worst-case update time $O(\log \Phi)$, where Φ is the *aspect ratio* of the point set X, defined as the ratio between the maximum and minimum pairwise distances in X. Krauthgamer and Lee [43] obtained similar results for proximity search, where the time bounds of their dynamic updates depend also in the aspect ratio. For proximity search, later, Cole and Gottlieb [26] removed the dependency on the aspect ratio. For the Euclidean plane, Abam et al. [1] presented an algorithm with $O(\log n)$ update time. For doubling metrics, Roditty [50] presented the first fully dynamic algorithm for maintaining a geometric $(1+\varepsilon)$-spanner, with update time depending only on the number of points in X. The algorithm obtained in [50] supports insertions in amortized $O(\log n)$ time and deletions in amortized $\tilde{O}(n^{1/3})$ time where $\tilde{O}$ hides polylogarithmic factors, and maintains spanner with $O(n/\varepsilon^d)$ edges. Gotllieb and Roditty [37] gave a dynamic spanner that supports insertions in $O\big(\varepsilon^{-O(d)} \log^2 n\big)$ amortized time and deletions in $O\big(\varepsilon^{-O(d)} \log^3 n\big)$ amortized time. Later, in [38],[1] they gave a $(1+\varepsilon)$-spanner of constant degree that supports updates in $\varepsilon^{-O(d)} \log n$ worst-case time.

Roughly speaking, these "classic" constructions of dynamic spanners for doubling metric are based on a *net tree* (see Section 2). While it is straightforward to construct a sparse spanner from a net tree, this tool is too rough to obtain lightness. Indeed, in Figure 1 we illustrate the net tree of the path graph, and show that it has lightness $\Omega(\log n)$. A more

[1] Gotllieb and Roditty [38] additionally assume that they can query the distance between a deleted point and a non-deleted point, which (for example) is possible for Euclidean metrics.

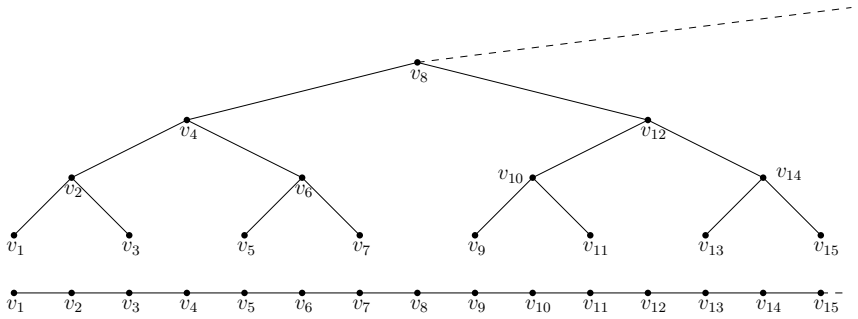


Figure 1 On bottom: The path graph P_n. All edges are unit weight. On top: a depiction of the net tree T of P_n. There is a laminar hierarchy of nets; the 2^i-net is $N_i = \{v_j \in P_n \mid j \mod 2^i = 0\}$ (all vertices at hight i in the tree and above). The net-tree spanner contains all the tree edges, and many additional edges. The weight of the edge $\{v_i, v_j\}$ is $|i-j|$. The tree T has depth $\log n$, and the total weight of the edges going from depth i vertices to depth $i+1$ is $\Theta(n)$. Thus $w(T) = \Theta(n \cdot \log n)$. As the weight of the MST of P_n is $n-1$, the net-tree spanner has lightness $\Omega(\log n)$.

recent approach to constructing dynamic sparse spanners is using *locality-sensitive orderings (LSO)* (introduced by Chan, Har-Peled, and Jones [22]). In the full version of the paper, we summarize the history of LSO, and prove that LSO-based spanners have logarithmic lightness (even for $\mathbb{R}^1$).

While there is a substantial body of work on dynamic spanners in Euclidean metrics and, more generally, in doubling metrics, as summarized above, there is comparatively little work on dynamic *light* spanners. Controlling lightness is arguably a significantly harder objective, as evidenced by the extensive literature in the offline setting (see, e.g., [45, 5, 49, 18, 44, 47, 13, 33, 36, 29, 14, 23] and the references therein).

▶ **Question 1.** *Can we design a dynamic algorithm that maintains a* light-weight *spanner for a point set under dynamic updates in doubling metrics with polylogarithmic update time?*

Partial progress toward Question 1 was made by Eppstein and Khodabandeh [31], who showed that one can maintain a *low-recourse* light spanner for dynamic point sets in low-dimensional Euclidean space. The *recourse* of a dynamic spanner is the number of edges that change after each insertion or deletion. They obtained a construction with amortized $O(1)$ recourse per insertion and amortized $O(\log \Phi)$ recourse per deletion. However, their result addresses recourse rather than efficiency: the actual time to update the spanner can be very large (no faster than just recomputing a spanner from scratch), and therefore their work does not resolve the question of achieving polylogarithmic (or even sublinear) update time.

1.1 Our Contribution

We resolve Question 1 in the affirmative. We say a set of points X is (a, b)-bounded if $\min_{u,v \in X} \delta(u, v) \geq a$ and $\max_{u,v \in X} \delta(u, v) \leq b$.

▶ **Theorem 2.** *Let $(\mathcal{X}, \delta)$ be a metric space with doubling dimension d, and let $\varepsilon > 0$ and $\Phi > 0$ be parameters. Let $X \subseteq \mathcal{X}$ be a set of points undergoing insertions and deletions, such that at all times X is $(1, \Phi)$-bounded. We maintain a $(1+\varepsilon)$-spanner L of X with lightness at most $\varepsilon^{-O(d)}$, where each insertion/deletion to X can be processed in time $(\varepsilon^{-1} \cdot \log \Phi)^{O(d)}$.*

For the sake of simplicity, we state our theorem as though the maximum distance Φ must be given in advance, and the minimum distance between points in X must be at least 1. Using standard techniques [43, 31] we could remove these restrictions at no loss in the running

time: we obtain a data structure whose update time depends adaptively on the aspect ratio Φ of the current point set X (see the full version of the paper). Additionally, we remark that we do not need to know the doubling dimension d in advance; the value d only appears in the runtime analysis.

Techniques. Our proof begin somewhat similarly to Eppstein and Khodabandeh [31]: roughly speaking, we seek to maintain a *greedy spanner* on top of a *dynamic sparse spanner* of points X (crucially, this spanner is much more structured than just the greedy spanner on the complete geometric graph on X). In Section 3, we first give a new (and arguably much simpler[2]) algorithm and analysis that such a spanner can be maintained with bounded recourse. After each insertion/deletion, we identify $O(\log \Phi)$ edges (u, v) of our greedy spanner L which might have to change, to satisfy the greedy spanner invariants: roughly speaking, we examine the distance $\delta_{L<}(u, v)$ between u and v in the graph consisting of all of edges of L that are shorter than (u, v)[3], and we insert or delete (u, v) to L based on whether $\delta_{L<}(u, v) \leq (1 + \varepsilon)\delta(u, v)$. Our key technical contribution (in Section 4) is to show that we can efficiently maintain estimates of $\delta_{L<}(u, v)$ even as we insert or delete edges into L.

1.2 Related Work

Dynamic Graph Spanners. In general graphs, the goal is to maintain, under edge updates, for any integer $k \geq 2$, a spanner with stretch $2k - 1$ and $\widetilde{O}(n^{1+1/k})$ edges. Assuming the Erdős–Girth conjecture, this trade-off is essentially optimal. The dynamic spanner problem was initiated by Ausiello et al. [7], and has since been studied extensively; see, for example, [8, 17, 27, 10, 34]. However, these works focus primarily on *sparsity*, and the resulting techniques do not naturally extend to maintaining *light* spanners. Moreover, due to inherent degree barriers in general graphs, this line of work exclusively considers *edge updates*. In contrast, in geometric and other structured metric spaces, vertex updates are often the more natural model, since the underlying graph need not be maintained explicitly. This distinction is crucial for light spanners, whose construction and analysis rely fundamentally on geometric structure rather than purely combinatorial sparsification.

Online Spanners. Spanners have also been studied extensively in the closely related *online* model. In this setting, points from a metric space arrive one by one, and at each step, the algorithm is given the subset of points that arrived so far. The goal is to maintain a t-spanner at all times. Unlike in the fully dynamic setting, the algorithm is allowed to *add* edges when a new point arrives, but may never remove previously added edges. In addition, the total number of points n is not known in advance. A number of algorithms and lower bounds have been obtained for this model, which addressed both sparsity and lightness guarantees along with stretch, under online constraints; see, e.g., [11, 15, 2, 3, 9, 40, 46]. These results achieve near-optimal trade-offs in the online regime and highlight the fundamental differences between incremental constructions and fully dynamic settings.

---

[2] We avoid the *bucketing* technique in the algorithm of [31] and instead work with all spanner edges at once; we maintain spanner invariants that work in doubling metrics and not just Euclidean metrics (unlike the *leapfrog property* used in [31]); and finally, we give a simple worst-case bound on the recourse of $O(\log \Phi)$, rather than using a potential function to bound the amortized recourse.

[3] Our description here assumes we want to maintain a greedy spanner, but actually we maintain a slightly more robust type of spanner which we call the *delayed greedy spanner*: we work with distances $\delta^*(u, v)$ rather than $\delta_{L<}(u, v)$, as defined in Section 2.2. We do this to control cascading changes to L.

2 Net trees and greedy spanners

Preliminaries. Throughout this paper, let $M = (\mathcal{X}, \delta)$ be a metric space with constant doubling dimension $d = O(1)$. For any vertex $x \in \mathcal{X}$ and radius $r > 0$, we let $B(x, r)$ denote the set of points within distance r of x; that is, $B(x, r) \coloneqq \{y \in \mathcal{X} : \delta(x, y) \leq r\}$. If G is a (geometric) graph, we use the notation $\delta_G(\cdot, \cdot)$ to denote shortest-path distances on G. For any positive integer n, we write $[n]$ to mean $\{i \in \mathbb{Z} : 0 \leq i \leq n\}$. Throughout this paper, we fix some sufficiently small positive integer $\varepsilon > 0$ and aim to construct a light $(1 + O(\varepsilon))$-spanner; our Theorem 2 will follow after rescaling $\varepsilon \leftarrow \varepsilon/O(1)$.

2.1 Net-Tree Spanners

In this subsection, we review the classic net-tree spanner [35].

Net tree. Let $X \subseteq \mathcal{X}$ be a set of points. Assume that the smallest distance between points in X is 1, and the largest distance is Φ; for simplicity we assume $\Phi = 2^i$ for some integer i. A *Δ-net* of X is a set of points N such that every point $x \in X$ is within distance Δ of some point in N, and any two net points in N are at distance greater than Δ. A Δ-net can be constructed greedily. A *net hierarchy* of X is a hierarchical sequence of nets $(N_0, N_1, \ldots, N_{\log \Phi})$ such that every N_i is a subset of N_{i-1}, and N_i is a 2^i-net of N_{i-1}; we take $N_0 = X$. Observe that (by a geometric series) every point in X is within distance 2^{i+1} of some net point in N_i. Moreover, the *packing bound* of doubling metrics implies that there are not too many points close together in each Δ-net.

▶ **Observation 3** (Packing bound). *Let N be a set of points that are pairwise at distance at least Δ. For any $\alpha \geq 1$ and $x \in \mathcal{X}$, there are at most $\alpha^{O(d)}$ points in $B(x, \alpha \cdot \Delta) \cap N$.*

An implicit representation of the net hierarchy $\mathcal{N}$ can be maintained using a data structure called the *net tree* [35, 43, 42]. We will sometimes abuse terminology and call $\mathcal{N}$ a net tree.

Net-tree spanner. The net-tree spanner was introduced (under the name "deformable spanner") by Gao, Guibas, and Nguyen [35]; similar ideas were independently designed by Krauthgamer and Lee [43]. The *ε-net-tree spanner* of X with respect to the net tree $\mathcal{N} = (N_0, \ldots, N_{\log \Phi})$ is the graph with vertex set X and with the following edge set: for every scale $i \in [\log \Phi]$, add an edge between any two vertices $u, v \in N_i$ if $\delta(u, v) \leq c \cdot 2^i$, where $c \coloneqq 4 + 16\varepsilon^{-1}$.

▶ **Lemma 4** ([35, Theorem 3.2]). *An ε-net-tree spanner of X is a $(1 + \varepsilon)$-spanner of X.*

For any pair of vertices (u, v), we say the *scale* of (u, v) is the value i such that $\delta(u, v) \in [2^{i-1}, 2^i)$. Observe that an edge (u, v) at scale i in an ε-net-tree spanner has both endpoints in the $O(\varepsilon) \cdot 2^i$-net of the net tree (not the 2^i-net). The packing bound implies that every vertex in the ε-net-tree spanner has degree at most $\varepsilon^{-O(d)} \cdot \log \Phi$. In fact, using a charging argument, one can show that the ε-net-tree spanner contains at most $\varepsilon^{-O(d)} \cdot n$ edges: that is, it is *sparse*. We frequently denote the ε-net-tree spanner as S.

Dynamic maintenance. Gao, Guibas, and Nguyen [35] showed that the net tree spanner can be maintained dynamically. We summarize their guarantee below.

▶ **Lemma 5** (Rephrasing of [35, Theorem 4.2]). *There is a data structure that implicitly maintains a hierarchy of nets $\mathcal{N} = (N_0, N_1, \ldots, N_{\log \Phi})$ of a point set X undergoing insertions and deletions. Each insertion or deletion can be processed in $O(\log \Phi)$ time. Moreover*[4]*, for any constant $\varepsilon > 0$, one can maintain an ε-net-tree spanner of X with respect to $\mathcal{N}$, in $O(\log \Phi)$ time per insertion or deletion.*

We will make use of Lemma 5, but we will have to slightly unwrap the black box – we will need some control on how the net tree $\mathcal{N}$ changes after each insertion or deletion. Fortunately, [35] update the net tree $\mathcal{N}$ in a simple and structured way. Inserting a point x into X has the following result, described in Figure 2.

$i_0 \leftarrow$ smallest scale such that the scale-i_0 net $N_{i_0} \in \mathcal{N}$ satisfies $\delta(x, N_{i_0}) < 2^{i_0}$
update $\mathcal{N}$ by inserting x into net $N_i \in \mathcal{N}$ for all $i < i_0$

Figure 2 Pseudocode for [35] algorithm to update net $\mathcal{N}$ after inserting point x into X.

Deleting a point x from X is described in Figure 3.

update $\mathcal{N}$ by deleting x from every net $N_i \in \mathcal{N}$
for every scale $i \leftarrow 1, 2, \ldots, \log \Phi$:
  while there exists some point $y \in N_{i-1}$ with $\delta(u, N_i) > 2^i$, add y to N_i

Figure 3 Pseudocode for [35] algorithm to update net $\mathcal{N}$ after deleting point x from X.

We will not discuss the details of the data structures or algorithms that [35] use to implement these two operations efficiently. We have unwrapped the [35] black box so that we can prove that every insertion/deletion of a point x only changes $\mathcal{N}$ "locally" nearby x.

▶ **Lemma 6.** *Let $\mathcal{N}^{\text{old}} = (N_0^{\text{old}}, \ldots, \mathcal{N}_{\log \Phi}^{\text{old}})$ be a net tree for point set X. Let $\mathcal{N} = (N_0, \ldots, N_{\log \Phi})$ be the net tree produced from $\mathcal{N}^{\text{old}}$ by the [35] algorithm after a point x is inserted into (resp. deleted from) X. For any scale $i \in [\log \Phi]$, every point y in the symmetric difference of N_i and N_i^{old} satisfies $\delta(y, x) \le 2^i$.*

Proof. When x is inserted into X (according to the pseudocode in Figure 2), the lemma follows trivially from the [35] algorithm description: the only point that could differ between N_i and N_i^{old} is x. We now argue that the lemma holds when x is deleted from X (according to the pseudocode in Figure 3). By construction, the only point that could be removed from N_i^{old} is x; that is, the set $N_i^{\text{old}} \setminus N_i$ is either $\varnothing$ or $\{x\}$. So it remains to show that every point y in $N_i \setminus N_i^{\text{old}}$ satisfies $\delta(y, x) \le 2^i$. The proof is by induction on i. When $i = 0$, every point in $\texttt{DS}.X$ is contained in N_0^{old}, so $N_0 \setminus N_0^{\text{old}} = \varnothing$ and the claim holds trivially. Now consider $i > 0$. Consider some point $y \in N_i \setminus N_i^{\text{old}}$. By the algorithm description, the point y satisfies

$$\delta(y, N_i^{\text{old}} \setminus \{x\}) > 2^i.$$

There are two cases. In the first case, suppose $y \in N_{i-1}^{\text{old}}$. Because N_i^{old} is a 2^i-net for N_{i-1}^{old}, we have $\delta(y, N_i^{\text{old}}) \le 2^i$. We conclude that $\delta(y, x) \le 2^i$ as desired. In the second case, suppose $y \notin N_{i-1}^{\text{old}}$; thus $y \in N_{i-1} \setminus N_{i-1}^{\text{old}}$, and by induction we have that $\delta(y, x) \le 2^{i-1} < 2^i$. ◀

[4] Note that the phrasing of [35, Theorem 4.2] does not explicitly separate the maintenance of the net hierarchy $\mathcal{N}$ from the maintenance of the spanner (they just state the existence of a dynamic spanner); however, our phrasing of Lemma 5 is immediate from their proof. In our application, we will need to dynamically maintain an ε_1-net-tree spanner of X and an ε_2-net-tree spanner of X (for two different values $\varepsilon_1 \neq \varepsilon_2$) *with respect to the same net tree* $\mathcal{N}$. This is possible using the data structure of [35].

We need one more guarantee from [35] (see full paper for details).

▷ Claim 7. Consider the data structure of [35] that maintains a hierarchy of nets $(N_0, N_1, \ldots, N_{\log \Phi})$ of X and an ε-net-tree spanner. Let $c_\varepsilon = 4 + 16\varepsilon^{-1}$. Given a scale $i \in [\log \Phi]$ and a query point $x \in \mathcal{X}$, one can find all vertices in $N_i \cap B(x, 100 \cdot c_\varepsilon \cdot 2^i)$ in time $\varepsilon^{-O(d)} = O(1)$.

2.2 Invariants for Our Light Spanner: Delayed Greedy Spanner

While the net-tree spanner S is sparse and can be maintained dynamically, it could have lightness $\Omega(\log n)$ (see Figure 1). We aim to find a subgraph of S with $O(1)$ lightness. Intuitively, we want to maintain a greedy $(1+\varepsilon)$-spanner H of the graph S: the greedy spanner (and even the *approximate* greedy spanner) are light [19, 33]. Recall that the greedy spanner initializes $H \leftarrow \varnothing$ and processes edges in S from smallest to largest; when we process (u, v), we add (u, v) to H iff $\delta_H(u, v) > (1+\varepsilon)\delta_S(u, v)$. In other words, the greedy spanner satisfies: for every edge (u, v) in S, letting $H_{<(u,v)}$ denote the set of edges in H with smaller[5] lengths than (u, v), we have $\delta_{H_{<(u,v)}}(u, v) > (1+\varepsilon)\delta_S(u, v)$ iff $(u, v) \in L$.

Delayed greedy spanner. We maintain a light spanner, denoted $S \subseteq S$ that doesn't obey the greedy spanner invariants exactly; rather, we define a more robust set of invariants. For any scale $i \in [\log \Phi]$, we define the set of edges $L[i] \subseteq L$ to be the set of all edges in L with *scale strictly less than* i; that is, every edge in $L[i]$ has length at most 2^{i-1}. For any edge (u, v) of the ε-net-tree spanner at scale i, we define $\delta^*(u, v) := \delta_{L[i]}(u, v)$; recall that $\delta_{L[i]}(\cdot, \cdot)$ denotes the shortest-path metric in the geometric graph induced by $L[i]$. To construct our light spanner, we maintain a subset of edges $L \subseteq S$ with the following properties:

▶ **Invariant 8.** *Every edge $(u, v) \in S \setminus L$ satisfies $\delta^*(u, v) \leq (1+\varepsilon) \cdot \delta(u, v)$.*

▶ **Invariant 9.** *Every edge $(u, v) \in L$ satisfies $\delta^*(u, v) > (1 + \frac{\varepsilon}{3}) \cdot \delta(u, v)$.*

If L satisfies Invariants 8 and 9, we call L a *delayed greedy spanner* of S (because there is a delay between the time an edge is added to the spanner and the time it is taken into account when making the next decision). We emphasize that the delayed greedy spanner invariants are different than the greedy spanner invariants for H: in our invariants, the choice of adding (u, v) to the spanner depends on $L[i]$ (the set of edges in L with *scale* strictly smaller than the scale of (u, v)), rather than $H_{<(u,v)}$ (the set of edges in the greedy spanner H that are strictly shorter than $\delta(u, v)$). Nevertheless, we show that any set of edges L that satisfies Invariants 8 and 9 is a $(1 + O(\varepsilon))$-spanner (Lemma 10) and has constant lightness (Lemma 11).

▶ **Lemma 10.** *Let S be an ε-net-tree spanner for X, and let $L \subseteq S$. If Invariant 8 holds, then L is a $(1+3\varepsilon)$-approximate spanner for (X, δ).*

Proof. By Lemma 4, the ε-net-tree spanner S is a $(1+\varepsilon)$-spanner for (X, δ). Moreover, Invariant 8 implies that L is a $(1+\varepsilon)$-spanner for S: by triangle inequality it suffices to check that every edge of S is preserved up to $1+\varepsilon$ approximation in L, and indeed, for every edge $(u, v) \in S$, we have $\delta_L(u, v) \leq \delta^*(u, v) \leq (1+\varepsilon)\delta(u, v) \leq (1+\varepsilon)\delta_S(u, v)$. We conclude that L is a spanner for (X, δ) with stretch $(1+\varepsilon)^2 \leq (1+3\varepsilon)$. ◀

[5] assume all edge lengths in S are distinct, by breaking ties arbitrarily

▶ **Lemma 11.** *Let S be an ε-net-tree spanner for X, and let $L \subseteq S$. If Invariants 8 and 9 hold, then L has lightness $\varepsilon^{-O(d)}$.*

Proof. We use, as a black box, the fact that the greedy $(1+\frac{\varepsilon}{3})$-spanner in a metric space of doubling dimension d has lightness $\varepsilon^{-O(d)}$ [19]. We consider the shortest-path metric δ_L induced by the edges of L; because (X, δ) has doubling dimension d, the stretch bound of Lemma 10 together with the definition of doubling dimension implies that the doubling dimension of (X, δ_L) is $O(d)$ (see [33, Observation 7]). Consider building the greedy $(1+\frac{\varepsilon}{3})$-spanner $\hat{L}$ of L: that is, we initialize $\hat{L} \leftarrow \varnothing$, then iterate over the edges of L from smallest to largest[6], and add an edge $(u,v) \in L$ into the spanner $\hat{L}$ iff $\delta_{\hat{L}}(u,v) > (1+\frac{\varepsilon}{3})\delta_L(u,v)$. The greedy spanner $\hat{L}$ has lightness $\varepsilon^{-O(d)}$. We now *charge* every edge of L to an edge of $\hat{L}$. If an edge $(u,v) \in L$ is added to $\hat{L}$, then charge (u,v) to itself. Otherwise, suppose $(u,v) \in L$ is not added to $\hat{L}$. This means that there is a path P between u and v in $\hat{L}$ (at the instant that (u,v) is considered) with length at most $(1+\frac{\varepsilon}{3})\delta_L(u,v)$. Because S is a geometric graph, $\delta_S(u,v) = \delta(u,v)$; i.e., P has length at most $(1+\frac{\varepsilon}{3})\delta(u,v)$. Let i be the scale of (u,v), meaning that $\delta(u,v) \in [2^{i-1}, 2^i)$. The path P contains some scale-i edge (u',v') in the greedy spanner $\hat{L}$: otherwise, if P contained only scale-j edges with $j < i$, we would have $\delta^*(u,v) \le (1+\frac{\varepsilon}{3}) \cdot \delta(u,v)$, contradicting Invariant 9. Charge (u,v) to the scale-i edge (u',v').

We now claim that every edge (u',v') in $\hat{L}$ is charged at most $\varepsilon^{-O(d)}$ times. Indeed, whenever a scale-i edge (u',v') is charged by a scale-i edge (u,v), we have $\delta(u,u') \le (1+\frac{\varepsilon}{3})2^i$; this is because, by definition of charging, u' lies on a path u and v with length at most $(1+\frac{\varepsilon}{3})\delta(u,v) \le (1+\frac{\varepsilon}{3})2^i$. The packing bound (Observation 3) implies that there are only $\varepsilon^{-O(d)}$ scale-i edges that could charge (u',v'). Because we only charge scale-i edges in L to scale-i edges in $\hat{L}$, we conclude that L is at most $\varepsilon^{-O(d)}$ times heavier than L, which is $\varepsilon^{-O(d)}$ times heavier than the minimum spanning tree on X. ◀

3 Light Spanner with Worst-Case Recourse Bounds

We now describe how to maintain our light spanner L, satisfying Invariants 8 and 9. The current section focuses on describing a spanner with a small *recourse* bound, and proving that our updates maintain Invariants 8 and 9. In Section 4, we explain how to perform these updates quickly. We maintain a data structure *DS*. It stores:

- a set of points *DS*.X
- a net-tree *DS*.$\mathcal{N}$ on the points DS.X
- an ε-net-tree spanner *DS*.S of DS.X with respect to DS.$\mathcal{N}$
- a subgraph *DS*.$L \subseteq$ DS.S called the *light spanner*, such that Invariants 8 and 9 hold.

All these variables are initialized as $\varnothing$. The data structure supports two operations – insertion and deletion of a point x. We say that a procedure Insert(x) *correctly implements insertion* of point x if the procedure assigns DS.$X \leftarrow$ DS.$X \cup \{x\}$ and updates DS.$\mathcal{N}$, DS.S, and DS.L to satisfy the descriptions above. Similarly, a procedure Delete(x) *correctly implement deletion* of a point x if it assigns DS.$X \leftarrow$ DS.$X \setminus \{x\}$ and updates DS.$\mathcal{N}$, DS.S, and DS.L.

3.1 Insertion

To insert a point x, we first update the net-tree according to [35] (as summarized in our Lemma 5) and then recompute the spanner edges DS.L nearby x to satisfy the invariants. The pseudocode is given below. Our key insight is that when we recompute spanner edges at level i, we only need to recompute edges with endpoints in $B(x, O(2^i))$.

[6] by triangle inequality, it suffices to examine only the edges of L when building the greedy spanner, rather than all pairs of vertices in L

```
DS.INSERT(x):
  DS.X ← DS.X ∪ {x}
  update DS.𝒩 and DS.S according to Lemma 5
  DS.RECOMPUTE(x)

DS.RECOMPUTE(x):
  for every scale i ← 0, 1, ..., log Φ:
      for every scale-i edge (u, v) of DS.S with u, v ∈ B(x, 4 · 2^i):
          if δ*(u, v) > (1 + ε) · δ(u, v):
              DS.L ← DS.L ∪ {(u, v)}
          else:
              DS.L ← DS.L \ {(u, v)}
```

▶ **Lemma 12.** *The procedure DS.INSERT(x) correctly implements the insertion of a point x.*

Proof. Let $\mathcal{N}^{\text{old}}$, S^{old}, L^{old}, and X^{old} denote $\texttt{DS}.\mathcal{N}$, $\texttt{DS}.S$, $\texttt{DS}.L$, and $\texttt{DS}.X$ (respectively) before the insertion process. For brevity, we write $\mathcal{N}$, S, L, and X instead of $\texttt{DS}.\mathcal{N}$, $\texttt{DS}.S$, $\texttt{DS}.L$, and $\texttt{DS}.X$. By Lemma 5, the variables $\texttt{DS}.X$, $\texttt{DS}.\mathcal{N}$ and $\texttt{DS}.S$ are correctly maintained. Moreover, we have that $L \subseteq S$, because (by description of the algorithm) $S^{\text{old}} \subseteq S$ and (by assumption) $L^{\text{old}} \subseteq S^{\text{old}}$.

It remains to show that that every edge (u, v) in S satisfies Invariants 8 and 9. Suppose that edge (u, v) is at scale i (that is, $\delta(u, v) \in (2^{i-1}, 2^i]$). Note that the distance $\delta^*(u, v) = \delta_{L[i]}(u, v)$ depends only on the edges in L with scale strictly less than i; thus, it suffices to show that (u, v) satisfies the two invariants at the moment after the for-loop processes scale i. If $u, v \in B(x, 4 \cdot 2^i)$, then (u, v) satisfies the two invariants by description of the algorithm. Now suppose that some endpoint, without loss of generality u, is not in $B(x, 4 \cdot 2^i)$. In this case, (u, v) is in L if and only if (u, v) is in L^{old}. There are two cases:

- **Suppose that (u, v) is not in L.** Thus (u, v) is not in L^{old}. To satisfy Invariant 8, we must show $\delta_{L[i]}(u, v) \leq (1+\varepsilon)\cdot\delta(u, v)$. In fact, we show that either $\delta_{L[i]}(u, v) \leq \delta_{L^{\text{old}}[i]}(u, v)$ or $\delta_{L^{\text{old}}[i]}(u, v) > 2\delta(u, v)$; see Figure 4. This suffices, as L^{old} satisfies Invariant 8: we conclude that $\delta_{L[i]}(u, v) \leq \delta_{L^{\text{old}}[i]}(u, v) \leq (1 + \varepsilon)\delta(u, v)$ as desired. Consider a shortest path P in $L^{\text{old}}[i]$ between u and v. The description of the algorithm implies that the only edges that that change between $L[i]$ and $L^{\text{old}}[i]$ are in the ball $B(x, 4 \cdot 2^{i-1})$. Because we assumed $\delta(u, x) > 4 \cdot 2^i$, triangle inequality implies that endpoints of any edge (u', v') in $L^{\text{old}}[i] \setminus L[i]$ are far from u: specifically, $\delta(u, u') \geq 4 \cdot 2^i - 4 \cdot 2^{i-1} = 2 \cdot 2^i$, and so $\delta_{L^{\text{old}}}(u, u') \geq 2 \cdot 2^i$. If P has length $\geq 2 \cdot 2^i$ then we are done; otherwise, P does not include any edges in $L^{\text{old}}[i] \setminus L[i]$, and so P is also a path in $L[i]$ as desired.
- **Suppose (u, v) is in L.** Thus (u, v) is in L^{old}. To satisfy Invariant 9, we must show that $\delta_{L[i]}(u, v) > (1 + \frac{\varepsilon}{3}) \cdot \delta(u, v)$. We claim that either $\delta_{L[i]}(u, v) \geq \delta_{L^{\text{old}}[i]}(u, v)$, or $\delta_{L[i]}(u, v) \geq 2\delta(u, v)$. This suffices to prove the claim, because L^{old} satisfies Invariant 9 and so $\delta_{L^{\text{old}}[i]}(u, v) > (1 + \frac{\varepsilon}{3})\delta(u, v)$; thus, in either case, $\delta_{L[i]}(u, v) > (1 + \frac{\varepsilon}{3})\delta(u, v)$. Let P be a shortest path between u and v in $L[i]$. If $\delta_{L[i]}(u, v) \geq \delta_{L^{\text{old}}[i]}(u, v)$ then we are done. Otherwise, if $\delta_{L[i]}(u, v) < \delta_{L^{\text{old}}[i]}(u, v)$, then P must contain some edge of $L[i]$ that is not in $L^{\text{old}}[i]$. But the description of the algorithm implies that any such edge has endpoints in $B(x, 4 \cdot 2^{i-1})$. By triangle inequality and our assumption that $\delta(u, x) \geq 4 \cdot 2^i$, we have $\delta(u, u') \geq 2 \cdot 2^i$, and so P has length at least $2 \cdot 2^i$ as desired. ◀

We remark that the INSERT procedure has bounded recourse: at most $\varepsilon^{-O(d)} \log \Phi$ edges change in the spanner L. This is because, for every scale $i \in \lceil \log \Phi \rceil$, there are at most $\varepsilon^{-O(d)}$ scale-i edges in the ε-net-tree spanner in $B(x, 4 \cdot 2^i)$, by definition of net-tree spanner and packing bound (Observation 3).

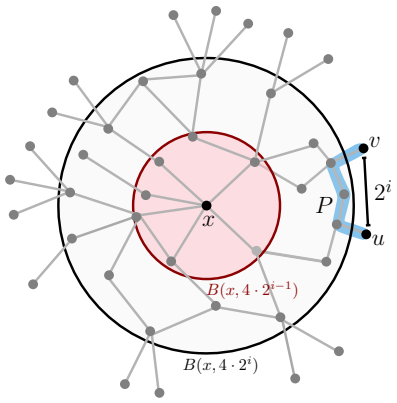


Figure 4 A light spanner L^{old}. For a given scale i, the spanners $L[i]$ and $L^{\text{old}}[i]$ differ only within the ball $B(x, 4 \cdot 2^{i-1})$. A scale-i pair (u, v) outside of $B(x, 4 \cdot 2^i)$. A short path P between u and v in L^{old} does not wander inside $B(x, 4 \cdot 2^{i-1})$, so P is also in L.

3.2 Deletion

The procedure for deleting a point x is similar to insertion. The procedure *DS*.DELETE(x) first updates the net tree according to [35], then deletes all edges in L that were incident to x, and finally recompute the spanner edges nearby x using the DS.RECOMPUTE procedure. The pseudocode and analysis are fairly similar to insertion procedure (albeit slightly more complicated). We defer all details to the full version of the paper.

▶ **Lemma 13.** *The procedure DS.DELETE(x) correctly implements the deletion of a point x.*

We pause to remark that in our proof of Lemma 12 (and also Lemma 13), we have actually proved the following claim, which will be helpful later.

▶ **Lemma 14.** *Consider the data structure DS before and after running DS.INSERT(x) or DS.DELETE(x). Let S^{old} (resp. L^{old}) denote DS.S (resp. DS.L) before the insertion/deletion is performed. Let (u, v) be a scale-i edge in DS.S, with some endpoint outside of $B(x, 4 \cdot 2^i)$. Then (u, v) is an edge in S^{old}, and either (1) $\delta_{DS.L[i]}(u, v) = \delta_{L^{\text{old}}[i]}(u, v)$, or (2) both $\delta_{DS.L[i]}(u, v)$ and $\delta_{L^{\text{old}}[i]}(u, v)$ are larger than $2\delta(u, v)$.*

4 Implementing Updates Quickly

In this section, we show how to modify the data structure so that the DS.INSERT and DS.DELETE procedures can be executed quickly. By prior work on the net-tree spanner (Lemma 5), the net-tree DS.$\mathcal{N}$ and the ε-net-tree spanner DS.S can be maintained in $O(\log \Phi)$ time. The DS.RECOMPUTE procedure iterates over each of the $\log \Phi$ scales. For each scale i, it examines the $\varepsilon^{-O(d)} \log \Phi$ many scale-i edges in $B(x, 4 \cdot 2^i)$; by Claim 7 these edges can be found in $\varepsilon^{-O(d)} = O(1)$ time. Each examined edge (u, v) is added or removed from DS.L depending on $\delta^*(u, v)$. This means that the bottleneck for a fast update time is the computation of $\delta^*(u, v)$:

▶ **Lemma 15.** *The procedure DS.INSERT and DS.DELETE run in $\varepsilon^{-O(d)} \cdot \tilde{O}(\log \Phi)$ time, plus the time needed for $\varepsilon^{-O(d)} \cdot \log \Phi$ many computations of $\delta^*(u, v)$.*

Unfortunately, we don't know how to design a data structure that maintains these distances. To get around this, we first observe that it suffices to maintain an *approximation* for $\delta^*(u,v)$.

▶ **Definition 16.** *We say a value $\tilde{d}$ is an α-approximation for a value d if $d \le \tilde{d} \le \alpha \cdot d$. For any function $d(u,v)$ on pairs of points (we will take $d(u,v) = \delta^*(u,v)$ and $d(u,v) = \delta_{\mathtt{DS}.L}(u,v)$), we say that $\tilde{d}(u,v)$ is a coarse α-approximation for $d(u,v)$ if:*

$$\tilde{d}(u,v) = \begin{cases} \text{if } d(u,v) \le 2\delta(u,v) \text{ then} & d(u,v) \le \tilde{d}(u,v) \le \alpha \cdot d(u,v) \\ \text{if } d(u,v) > 2\delta(u,v) \text{ then} & \tilde{d}(u,v) \ge 2 \cdot \delta(u,v) \end{cases}$$

In other words, a coarse α-approximation $\tilde{d}(u,v)$ for $\delta^*(u,v)$ lets us either detect when $\delta^*(u,v)$ is very large or provides an α-approximation for $\delta^*(u,v)$. Throughout this section, we set $\alpha := 1 + \frac{\varepsilon}{3}$. As we show in Lemma 19, even if we used coarse α-approximations $\tilde{\delta}^*(u,v)$ instead of the real distances $\delta^*(u,v)$ in the $\mathtt{DS}$.RECOMPUTE procedure, our spanner L still satisfies Invariants 8 and 9 after the $\mathtt{DS}$.INSERT and $\mathtt{DS}$.DELETE procedures.

Augmenting the data structure with distance estimates. We now show that we can modify our data structure to efficiently maintain coarse α-approximations of $\delta^*(u,v)$ for every edge (u,v) in the ε-net-tree $\mathtt{DS}.S$. In fact, we maintain something stronger: we need to store several auxiliary values in order to maintain the coarse approximations. Let $\kappa \ge \frac{1024}{3}$ be a sufficiently large constant, and let $\varepsilon_{\text{small}} \leftarrow \frac{\varepsilon}{3 \cdot \kappa \cdot \log \Phi}$. Our data structure $\mathtt{DS}$ maintains (in addition to the points $\mathtt{DS}.X$, the net-tree $\mathtt{DS}.\mathcal{N}$, the ε-net-tree spanner $\mathtt{DS}.S$, and the light spanner $\mathtt{DS}.L$) the following objects:

- An $\varepsilon_{\text{small}}$-net tree spanner $\mathtt{DS}.S_{\text{small}}$ for $\mathtt{DS}.\mathcal{N}$
- For every scale $i \in [\log \Phi]$ and every edge (u,v) in $\mathtt{DS}.S_{\text{small}}$ at scale i, a value $\mathtt{DS}.\tilde{\delta}^*(u,v)$ which is a *coarse* $(1 + \kappa \cdot i \cdot \varepsilon_{\text{small}})$-approximation for $\delta^*(u,v)$.
- For every scale $i \in [(\log \Phi) - 2]$ and every edge (u,v) in $\mathtt{DS}.S_{\text{small}}$ at scale i, a value $\mathtt{DS}.\tilde{\delta}_L(u,v)$ which is a $(1 + \kappa \cdot i \cdot \varepsilon_{\text{small}})$-approximation for $\delta_{\mathtt{DS}.L}(u,v)$.

Observe that this data structure does indeed maintain a coarse $\frac{\varepsilon}{3}$-approximation for every edge in $\mathtt{DS}.S$: this is because $\mathtt{DS}.S \subset \mathtt{DS}.S_{\text{small}}$, and $\kappa \cdot i \cdot \varepsilon_{\text{small}} < \frac{\varepsilon}{3}$ so the estimates $\mathtt{DS}.\tilde{\delta}^*(u,v)$ are coarse $\frac{\varepsilon}{3}$-approximations for $\delta^*(u,v)$. The reason we maintain estimates for every edge in $\mathtt{DS}.S_{\text{small}}$ and not just every edge in $\mathtt{DS}.S$ is to control the distortion, which will accumulate by a $(1 + O(\varepsilon_{\text{small}}))$ factor at each scale.

We highlight two differences between the estimates $\mathtt{DS}.\tilde{\delta}^*(u,v)$ and $\mathtt{DS}.\tilde{\delta}_L(u,v)$, for a scale-$i$ edge (u,v). First, the value $\mathtt{DS}.\tilde{\delta}^*(u,v)$ seeks to estimate $\delta^*(u,v)$, which is the distance between u and v in the subgraph $\mathtt{DS}.L[i]$ which consists of all edges in $\mathtt{DS}.L$ with scale strictly smaller than i. On the other hand, $\mathtt{DS}.\tilde{\delta}_L(u,v)$ estimates distances in the graph $\mathtt{DS}.L$; note that the shortest path between u and v in $\mathtt{DS}.L$ may include edges in $\mathtt{DS}.L$ at scale i and $i+1$, in addition to edges in $\mathtt{DS}.L[i]$. The second key difference is that $\mathtt{DS}.\tilde{\delta}^*(u,v)$ is only required to be a $(1 + \kappa \cdot i \cdot \varepsilon_{\text{small}})$-approximation of $\delta^*(u,v)$ when $\delta^*(u,v) \le 2\delta(u,v)$, whereas $\mathtt{DS}.\tilde{\delta}_L(u,v)$ is always a $(1 + \kappa \cdot i \cdot \varepsilon_{\text{small}})$-approximation of $\delta_{\mathtt{DS}.L}(u,v)$. Why do we need these two types of estimates? Ultimately, our algorithm for INSERT and DELETE will only use the estimates $\mathtt{DS}.\tilde{\delta}^*(u,v)$, to determine whether or not the edge (u,v) should be added. But in order to compute $\mathtt{DS}.\tilde{\delta}^*(u,v)$ for some pair (u,v) at scale i, we will need to estimate $\delta_{\mathtt{DS}.L}(u',v')$ for pairs (u',v') at scale $\le i - 3$.

The fast insert/delete procedures. Below, we give pseudocode for a modified DS.RECOMPUTE procedure called *DS.RECOMPUTEFAST*, which uses the estimates $\mathtt{DS}.\tilde{\delta}^*(u,v)$ instead of the true distances $\delta^*(u,v)$. We define the procedures *DS.INSERTFAST* and *DS.DELETEFAST* to be identical to DS.INSERT and DS.DELETE, except that they use DS.RECOMPUTEFAST instead of DS.RECOMPUTE, and they also maintain a $\varepsilon_{\text{small}}$-net-tree spanner $\mathtt{DS}.S_{\text{small}}$ in addition to the ε-net-tree spanner $\mathtt{DS}.S$.

```
DS.RecomputeFast(x):
  for every scale i ← 0, 1, ..., log φ:
      DS.UpdateDistEstimates(x, i) ⟨⟨recomputes estimates DS.δ̃*(·,·) and DS.δ̃_L(·,·).⟩⟩
      for every scale-i edge (u, v) of DS.S with u, v ∈ B(x, 8 · 2^i):
          d̃ ← DS.δ̃*(u, v) ⟨⟨a ε/3-good estimate for δ*(u, v)⟩⟩
          if d̃ > (1 + ε) · δ(u, v):
              DS.L ← DS.L ∪ {(u, v)}
          else:
              DS.L ← DS.L \ {(u, v)}
```

In Section 4.1, we describe the procedure DS.UPDATEDISTESTIMATES and prove that it correctly maintains estimates for $\delta^*(u,v)$ and $\delta_L(u,v)$. For now, we just state lemmas which summarize the guarantees of DS.UPDATEDISTESTIMATES (Lemmas 17 and 18). Assuming the lemmas hold, we prove the correctness of the fast insert/delete procedures (Lemma 19).

▶ **Lemma 17.** *DS.UPDATEDISTESTIMATES runs in time* $(\varepsilon^{-1} \cdot \log \Phi)^{O(d)}$.

▶ **Lemma 18.** *Consider the execution of DS.INSERTFAST*(x) *or DS.DELETEFAST*(x). *Let* $i \in [\log \Phi]$ *be a scale, and consider the data structure DS at the instant after the execution of the call DS.UPDATEDISTESTIMATES*(x,i) *during the insertion/deletion process. Assume that every edge* (u,v) *of DS.S with scale strictly smaller than* i *satisfies Invariants 8 and 9. Then:*

- *for every scale-*i *edge* (u,v) *in the* $\varepsilon_{\text{small}}$*-net-tree spanner* $\mathtt{DS}.S_{\text{small}}$, *the value* $\mathtt{DS}.\tilde{\delta}^*(u,v)$ *is a coarse* $(1+\kappa \cdot i \cdot \varepsilon_{\text{small}})$*-approximation for* $\delta^*(u,v)$.
- *for every scale-*$(i-2)$ *edge* (u,v) *in the* $\varepsilon_{\text{small}}$*-net-tree spanner* $\mathtt{DS}.S_{\text{small}}$, *the value* $\mathtt{DS}.\tilde{\delta}_L(u,v)$ *is a* $(1+\kappa \cdot i \cdot \varepsilon_{\text{small}})$*-approximation* $\delta_{\mathtt{DS}.L}(u,v)$.

▶ **Lemma 19.** *The procedure DS.INSERTFAST*(x) *(resp. DS.DELETEFAST*(x)*) correctly implements insertion (resp. deletion) of* x. *It runs in time* $(\varepsilon^{-1} \cdot \log \Phi)^{O(d)}$.

Proof. The runtime bound follows from Lemmas 15 and 17. The proof of correctness is almost identical to that of Lemmas 12 and 13 except in one place: we need to argue that every scale-i edge (u,v) in $\mathtt{DS}.S$ with both endpoints in $B(x, 4 \cdot 2^i)$ satisfies Invariants 8 and 9.

For every such edge (u,v), the algorithm checks if $\mathtt{DS}.\tilde{\delta}^*(u,v) > (1+\varepsilon)\delta(u,v)$ and includes (u,v) in L iff the inequality holds. At the instant the algorithm makes the check, we may assume (by induction) that every edge (u',v') in $\mathtt{DS}.S$ with scale strictly smaller than i satisfies Invariants 8 and 9. Thus, by Lemma 18, the value of $\mathtt{DS}.\tilde{\delta}(u,v)$ is a coarse $(1+\kappa \cdot i \cdot \varepsilon_{\text{small}})$-approximation for $\delta^*(u,v)$. As $i \in [\log \Phi]$, in particular we conclude that $\mathtt{DS}.\tilde{\delta}(u,v)$ is a coarse $(1+\frac{\varepsilon}{3})$-approximation for $\delta^*(u,v)$. There are two cases.

Case 1: $\mathtt{DS}.\tilde{\delta}(u,v) \le (1+\varepsilon)\delta(u,v)$. In this case, (u,v) is not in $\mathtt{DS}.L$. We claim that $\delta^*(u,v) \le (1+\varepsilon)\delta(u,v)$, so that (u,v) satisfies Invariant 8. Indeed, we have $\delta^*(u,v) < 2\delta(u,v)$; otherwise, the definition of coarse α-approximation means that $\mathtt{DS}.\tilde{\delta}(u,v) > 2\delta(u,v)$, which contradicts our assumption that $\mathtt{DS}.\tilde{\delta}(u,v) \le (1+\varepsilon)\delta(u,v)$. Thus, the definition of coarse α-approximation implies $\delta^*(u,v) \le \mathtt{DS}.\tilde{\delta}(u,v)$ and so $\delta^*(u,v) \le (1+\varepsilon)\delta(u,v)$ as desired.

Case 2: $\texttt{DS}.\tilde{\delta}(u,v) > (1+\varepsilon)\delta(u,v)$. In this case, (u,v) is added to $\texttt{DS}.L$. We claim that $\delta^*(u,v) > (1+\frac{\varepsilon}{3})\delta(u,v)$, so that (u,v) satisfies Invariant 9. For contradiction, suppose that $\delta^*(u,v) \leq (1+\frac{\varepsilon}{3})\delta(u,v)$. The definition of coarse $(1+\frac{\varepsilon}{3})$-approximation implies $\texttt{DS}.\tilde{\delta}(u,v) \leq (1+\frac{\varepsilon}{3})\cdot\delta^*(u,v)$. Combining these two inequalities, we conclude that $\texttt{DS}.\tilde{\delta}(u,v) \leq \left(1+\frac{\varepsilon}{3}\right)\left(1+\frac{\varepsilon}{3}\right)\delta(u,v) \leq (1+\varepsilon)\delta(u,v)$, contradicting the Case 2 assumption. ◀

Theorem 2 follows from Lemmas 19, 10, and 11 (after rescaling $\varepsilon \leftarrow \varepsilon/3$).

4.1 The procedure UpdateDistEstimates

We defer the details of UpdateDistEstimates to the full version of our paper.

5 Conclusion and Open Questions

We have designed the first dynamic light spanner for point sets in Euclidean and doubling metrics, with update time $(\log \Phi)^{O(d)}$. Our work leaves open two interesting open questions. First, is it possible to maintain a dynamic light spanner with recourse or time bounds *independent* of the aspect ratio Φ and instead depends only on the number of points $n = |X|$? Second, can we obtain a runtime bound where the exponent of $\log \Phi$ (or $\log n$) is independent of d? Ideally, we would like to handle updates in time $\varepsilon^{-O(d)} \cdot \log n$, which would match the best runtime for dynamic sparse spanners.

References

1. Mohammad Ali Abam, Mark de Berg, and Joachim Gudmundsson. A simple and efficient kinetic spanner. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 306–310, 2008. doi:10.1145/1377676.1377729.
2. Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. A general approach to online network optimization problems. *ACM Trans. Algorithms*, 2(4):640–660, 2006. doi:10.1145/1198513.1198522.
3. Noga Alon and Yossi Azar. On-line steiner trees in the euclidean plane. In David Avis, editor, *Proceedings of the Eighth Annual Symposium on Computational Geometry, Berlin, Germany, June 10-12, 1992*, pages 337–343. ACM, 1992. doi:10.1145/142675.142744.
4. Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993. doi:10.1007/BF02189308.
5. Sunil Arya, Gautam Das, David M Mount, Jeffrey S Salowe, and Michiel Smid. Euclidean spanners: short, thin, and lanky. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 489–498, 1995. doi:10.1145/225058.225191.
6. Sunil Arya, David M Mount, and Michiel Smid. Dynamic algorithms for geometric spanners of small diameter: Randomized solutions. *Computational Geometry*, 13(2):91–107, 1999. doi:10.1016/S0925-7721(99)00014-0.
7. Giorgio Ausiello, Paolo G Franciosa, and Giuseppe F Italiano. Small stretch spanners on dynamic graphs. In Gerth Stølting Brodal and Stefano Leonardi, editors, *13th Annual European Symposium (ESA 2005)*, volume 3669 of *Lecture Notes in Computer Science*, pages 532–543. Springer, Springer, 2005. doi:10.1007/11561071_48.
8. Surender Baswana, Sumeet Khurana, and Soumojit Sarkar. Fully dynamic randomized algorithms for graph spanners. *ACM Transactions on Algorithms (TALG)*, 8(4):1–51, 2012. doi:10.1145/2344422.2344425.
9. Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 344–353. ACM, 1997. doi:10.1145/258533.258618.

10 Aaron Bernstein, Sebastian Forster, and Monika Henzinger. A deamortization approach for dynamic spanner and dynamic maximal matching. *ACM Transactions on Algorithms (TALG)*, 17(4):1–51, 2021. doi:10.1145/3469833.

11 Sujoy Bhore, Arnold Filtser, Hadi Khodabandeh, and Csaba D. Tóth. Online spanners in metric spaces. *SIAM J. Discret. Math.*, 38(1):1030–1056, 2024. doi:10.1137/22m1534572.

12 Sujoy Bhore, Balázs Keszegh, Andrey Kupavskii, Hung Le, Alexandre Louvet, Dömötör Pálvölgyi, and Csaba D. Tóth. Spanners in planar domains via steiner spanners and non-steiner tree covers. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4292–4326. SIAM, 2025. doi:10.1137/1.9781611978322.145.

13 Sujoy Bhore and Lazar Milenkovic. Light spanners with small hop-diameter. In *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, Aarhus, Denmark, July 8-11, 2025*, volume 334 of *LIPIcs*, pages 30:1–30:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.30.

14 Sujoy Bhore and Csaba D. Tóth. Euclidean steiner spanners: Light and sparse. *SIAM J. Discret. Math.*, 36(3):2411–2444, 2022. doi:10.1137/22m1502707.

15 Sujoy Bhore and Csaba D. Tóth. Online euclidean spanners. *ACM Trans. Algorithms*, 21(1):5:1–5:22, 2025. doi:10.1145/3681790.

16 Greg Bodwin. An alternate proof of near-optimal light spanners. In Merav Parter and Seth Pettie, editors, *2024 Symposium on Simplicity in Algorithms, SOSA 2024, Alexandria, VA, USA, January 8-10, 2024*, pages 39–55. SIAM, 2024. doi:10.1137/1.9781611977936.5.

17 Greg Bodwin and Sebastian Krinninger. Fully dynamic spanners with worst-case update time. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *LIPIcs*, pages 17:1–17:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.17.

18 Glencora Borradaile, Hung Le, and Christian Wulff-Nilsen. Minor-free graphs have light spanners. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 767–778. IEEE, 2017. doi:10.1109/FOCS.2017.76.

19 Glencora Borradaile, Hung Le, and Christian Wulff-Nilsen. Greedy spanners are optimal in doubling metrics. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2371–2379. SIAM, 2019. doi:10.1137/1.9781611975482.145.

20 Prosenjit Bose, Joachim Gudmundsson, and Pat Morin. Ordered theta graphs. *Computational Geometry*, 28(1):11–18, 2004. doi:10.1016/J.COMGEO.2004.01.003.

21 T-H Hubert Chan and Anupam Gupta. Small hop-diameter sparse spanners for doubling metrics. *Discrete & Computational Geometry*, 41(1):28–44, 2009. doi:10.1007/S00454-008-9115-5.

22 Timothy M Chan, Sariel Har-Peled, and Mitchell Jones. On locality-sensitive orderings and their applications. *SIAM Journal on Computing*, 49(3):583–600, 2020. doi:10.1137/19M1246493.

23 Hsien-Chih Chang, Jonathan Conroy, Hung Le, Shay Solomon, and Cuong Than. Light tree covers, routing, and path-reporting oracles via spanning tree covers in doubling graphs. In *STOC'25—Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 2257–2268. ACM, 2025. doi:10.1145/3717823.3718312.

24 Shiri Chechik and Christian Wulff-Nilsen. Near-optimal light spanners. *ACM Transactions on Algorithms (TALG)*, 14(3):1–15, 2018. doi:10.1145/3199607.

25 Vincent Cohen-Addad, Arnold Filtser, Philip N Klein, and Hung Le. On light spanners, low-treewidth embeddings and efficient traversing in minor-free graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 589–600. IEEE, 2020. doi:10.1109/FOCS46700.2020.00061.

26 Richard Cole and Lee-Ad Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 574–583, 2006. doi:10.1145/1132516.1132599.

27 Michael Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Transactions on Algorithms (TALG)*, 7(2):1–17, 2011. doi:10.1145/1921659.1921666.

28 Michael Elkin, Ofer Neiman, and Shay Solomon. Light spanners. *SIAM J. Discret. Math.*, 29(3):1312–1321, 2015. doi:10.1137/140979538.

29 Michael Elkin and Shay Solomon. Steiner shallow-light trees are exponentially lighter than spanning ones. *SIAM J. Comput.*, 44(4):996–1025, 2015. doi:10.1137/13094791X.

30 Michael Elkin and Shay Solomon. Fast constructions of lightweight spanners for general graphs. *ACM Trans. Algorithms*, 12(3):29:1–29:21, 2016. doi:10.1145/2836167.

31 David Eppstein and Hadi Khodabandeh. Maintaining light spanners via minimal updates. In *Proceedings of the 36th Canadian Conference on Computational Geometry (CCCG)*, pages 1–15, 2024.

32 Arnold Filtser and Ofer Neiman. Light spanners for high dimensional norms via stochastic decompositions. *Algorithmica*, 84(10):2987–3007, 2022. doi:10.1007/s00453-022-00994-0.

33 Arnold Filtser and Shay Solomon. The greedy spanner is existentially optimal. *SIAM J. Comput.*, 49(2):429–447, 2020. doi:10.1137/18M1210678.

34 Sebastian Forster and Gramoz Goranci. Dynamic low-stretch trees via dynamic low-diameter decompositions. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 377–388, 2019. doi:10.1145/3313276.3316381.

35 Jie Gao, Leonidas J. Guibas, and An Thai Nguyen. Deformable spanners and applications. *Comput. Geom.*, 35(1-2):2–19, 2006. doi:10.1016/j.comgeo.2005.10.001.

36 Lee-Ad Gottlieb. A light metric spanner. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 759–772. IEEE, 2015. doi:10.1109/FOCS.2015.52.

37 Lee-Ad Gottlieb and Liam Roditty. Improved algorithms for fully dynamic geometric spanners and geometric routing. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 591–600. SIAM, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347148.

38 Lee-Ad Gottlieb and Liam Roditty. An optimal dynamic spanner for doubling metric spaces. In *European Symposium on Algorithms*, pages 478–489. Springer, 2008. doi:10.1007/978-3-540-87744-8_40.

39 Michelangelo Grigni and Papa Sissokho. Light spanners and approximate tsp in weighted graphs with forbidden minors. In *SODA*, volume 2, pages 852–857, 2002. URL: http://dl.acm.org/citation.cfm?id=545381.545492.

40 MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Online node-weighted steiner forest and extensions via disk paintings. *SIAM J. Comput.*, 46(3):911–935, 2017. doi:10.1137/14098692X.

41 Sariel Har-Peled, Piotr Indyk, and Anastasios Sidiropoulos. Euclidean spanners in high dimensions. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 804–809. SIAM, 2013. doi:10.1137/1.9781611973105.57.

42 Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006. doi:10.1137/S0097539704446281.

43 Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 798–807. ACM, New York, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982913.

44 Hung Le and Shay Solomon. A unified framework for light spanners. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 295–308. ACM, 2023. doi:10.1145/3564246.3585185.

45 Hung Le and Shay Solomon. Truly optimal euclidean spanners. *SIAM J. Comput.*, 54(4):S19–135, 2025. doi:10.1137/20m1317906.

46 Joseph Naor, Debmalya Panigrahi, and Mohit Singh. Online node-weighted steiner tree and related problems. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 210–219. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.65.

47 Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, Cambridge, 2007. doi:10.1017/CBO9780511546884.

48 David Peleg and Alejandro A Schäffer. Graph spanners. *Journal of graph theory*, 13(1):99–116, 1989. doi:10.1002/JGT.3190130114.

49 Satish Rao and Warren D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 540–550. ACM, 1998. doi:10.1145/276698.276868.

50 Liam Roditty. Fully dynamic geometric spanners. *Algorithmica*, 62(3):1073–1087, 2012. doi:10.1007/S00453-011-9504-7.

51 Michiel HM Smid. The weak gap property in metric spaces of bounded doubling dimension. *Efficient Algorithms*, 5760:275–289, 2009. doi:10.1007/978-3-642-03456-5_19.

Improved Online Hitting Set Algorithms for Structured and Geometric Set Systems

Sujoy Bhore
Department of Computer Science and Engineering, Indian Institute of Technology Bombay, India

Anupam Gupta
Computer Science Department, New York University, NY, USA

Amit Kumar
Department of Computer Science and Engineering, Indian Institute of Technology Delhi, India

Abstract

In the online hitting set problem, sets arrive over time, and the algorithm has to maintain a subset of elements that hit all the sets seen so far. Alon, Awerbuch, Azar, Buchbinder, and Naor (SICOMP 2009) gave an algorithm with competitive ratio $O(\log n \log m)$ for the (general) online hitting set and set cover problems for m sets and n elements; this is known to be tight for efficient online algorithms. Given this barrier for general set systems, we ask: *can we break this double-logarithmic phenomenon for online hitting set/set cover on structured and geometric set systems?*

We provide an $O(\log n \log\log n)$-competitive algorithm for the weighted online hitting set problem on set systems with linear *shallow-cell complexity*, replacing the double-logarithmic factor in the general result by effectively a single logarithmic term. As a consequence of our results we obtain the first bounds for weighted online hitting set for natural geometric set families, thereby answering open questions regarding the gap between general and geometric weighted online hitting set problems.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Online algorithms

Keywords and phrases Hitting Set, Online Algorithms, Shallow-Cell Complexity, VC-Dimension

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.14

Related Version *Full Version*: `https://arxiv.org/abs/2603.14293` [6]

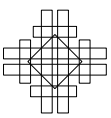
Funding *Anupam Gupta*: Supported in part by NSF awards CCF-2422926 and CCF-2608359.

1 Introduction

In the *hitting set problem*, we are given a set system $\mathcal{I} = (U, \mathcal{F})$, with ground set U and a collection of subsets $\mathcal{F}$ of U; every element e has a cost c_e. The goal is to select a minimum-cost subset $S \subseteq U$ that hits every set in $\mathcal{F}$. This problem has been a central one in algorithm design because of its many applications, and also because it serves as a test-bed for new techniques. The hitting set problem is equivalent to the *set cover* problem on a "dual" set system $\mathcal{I}^*$, which contains an element e_S for each original set $S \in \mathcal{F}$, and a set S_e for each element $e \in U$, such that $e \in S \iff e_S \in S_e$; i.e., we flip the roles of U and $\mathcal{F}$. Henceforth, we focus on hitting set.

In the offline setting, the hitting set (or the set cover) problem has been widely studied, with $(1 \pm o(1))\ln|U|$-approximation algorithms in the worst-case, which are best-possible if $P \neq NP$ [20]. In addition, hitting set has also been extensively studied for special set systems – most notably, those with low *VC-dimension* or bounded *shallow cell complexity (SCC)* [12, 7, 18, 11, 10, 26, 8], which result in constant-factor approximations for set systems like pseudo-disks and half-spaces in $\mathbb{R}^2$. These measures of set complexity arise not only in geometric and combinatorial settings, but in non-geometric problems like job scheduling [4].


© Sujoy Bhore, Anupam Gupta, and Amit Kumar;
licensed under Creative Commons License CC-BY 4.0



The progress for structured set systems in the *online* setting – where sets arrive over time, and must be hit as they arrive – has been more limited. While results are known for online hitting set (and set cover) for specific problems like intervals in $\mathbb{R}$ or 2-dimensional squares (see [18, 21]), general techniques are not known. The *offline* algorithms use quasi-uniform sampling procedures [26, 8] or black-box algorithms to find ε-nets [18], which have seemed difficult to extend to the online setting. As far as we know, the results for geometric instances have been restricted to the *unweighted* versions of the problems, and the $O(\log n \log m)$-competitiveness of [2] (where $n = |U|$ and $m = |\mathcal{F}|$) remains the only general result for weighted online hitting set, even for set systems with bounded VC dimension/SCC. We ask:

Can we achieve better competitive ratios for broad classes of set systems? Specifically, can we match the performance gains seen in structured offline settings?

1.1 Our Results

Our main results answer this question positively. Recall that the VC dimension of a set system is the largest set of elements that can be shattered. The *shallow cell complexity* $\varphi_{\mathcal{F}}(\cdot,\cdot)$ is a finer-grained notion, bounding the number of sets of size k in any restriction of the set system to ℓ elements.

The Unweighted Case. For unit-cost instances, we adapt the offline Net-Finder algorithm [22] to the online setting to give improved algorithm for set systems with small shallow cell complexity (SCC) and VC dimension:

▶ **Theorem 1** (Unweighted Hitting Set). *There is a randomized online algorithm for (unweighted) hitting set that has competitive ratio* $O(\log n \cdot (d + \log(d \cdot \varphi(d\,\mathsf{OPT}\,\log n, d))))$*, where* $\varphi(\cdot,\cdot)$ *is the shallow cell complexity (SCC) and* d *the VC dimension of the set system.*

We mention two corollaries of this work:

(i) For set systems with **"linear" SCC** (i.e., where $\varphi(\ell, k) = \text{poly}(k)$) and constant VC-dimension, the competitive ratio becomes $O(\log n)$. This gives hitting set and set cover algorithms for intervals in $\mathbb{R}$, lines and pseudo-disks – which includes disks and axis-aligned squares – in $\mathbb{R}^2$, and half-spaces in $\mathbb{R}^3$. (Table 1 gives the relevant bounds.) This result unifies and extends prior results for intervals, unit disks, and half-spaces [19], as well as for axis-aligned squares in $\mathbb{Z}^2$ [21]. This resolves the open problem [19, §8, Prob. #1] regarding hitting sets for arbitrary (not necessarily unit) disks in $\mathbb{R}^2$ (see also [21, §6, Prob. #1]), and another open problem in [21, §6, Prob. #5] which asks for results for hitting squares using non-integral point sets.

(ii) For set systems with **VC-dimension d**, we have $\varphi(\ell, k) \leq \ell^d$, and hence the competitive ratio becomes $O(d \log n \cdot \log(d\mathsf{OPT} \log n))$. Comparing this to the offline bound of $O(d \log(d \cdot \mathsf{OPT}))$, our online algorithm incurs only a near-logarithmic overhead.

The Weighted Case. If elements have arbitrary non-negative costs, we do not know of any prior online algorithms, so our result extending [8] to the online setting gives the first algorithm for structured set systems in this weighted case. Here, we state a special case of our result – for the "linear" case where $\varphi(\ell, k) = \text{poly}(k)$ – and defer the more detailed result to Theorem 16.

▶ **Theorem 2** (Weighted Hitting Set). *There is a randomized online algorithm for weighted hitting set that has competitive ratio* $O(\log n \log\log n)$ *for any set system with linear SCC.*

Table 1 Bounds on the SCC and VC dimensions of some primal/dual set systems from [23, 24]. Primal bounds give results for corresponding the hitting set problem, and dual bounds give results for the corresponding set cover problem. A triangle is α-fat if the ratio of the radius of the smallest enclosing ball to that of the largest inscribed ball is at most α.

Objects	Primal		Dual	
	$\varphi(\ell, k)$	VC-dim	$\varphi(\ell, k)$	VC-dim
Intervals in $\mathbb{R}$	$O(k)$	2	$O(1)$	2
Lines in $\mathbb{R}^2$	$O(k)$	2	$O(k)$	2
Pseudo-disks in $\mathbb{R}^2$	$O(k)$	3	$O(k)$	$O(1)$
α-Fat triangles in $\mathbb{R}^2$			$O(k \log^* {}^m/_k + {}^k/_\alpha \log^2 {}^1/_\alpha)$	7
Half-spaces in $\mathbb{R}^3$	$O(k^2)$	4	$O(k^2)$	4

This immediately extends the geometric results listed above in i to the weighted setting; to the best of our knowledge, these are the first results for weighted hitting set for these geometric instances. Our results for axis-aligned squares answers an open problem about weighted hitting sets from [21, §6, Prob. #4].

In both the unweighted and weighted cases, we lose nearly-logarithmic factors over the offline results: such a "price of uncertainty" is in line with typical losses for online vs. offline set cover. Moreover, such a logarithmic loss is unavoidable for the online setting, even for very structured set systems: [19] show a lower bound of $\Omega(\log n)$ for online hitting intervals on a line; observe that we can get exact offline algorithms for this problem.

1.2 Our Techniques

We demonstrate that online relax-and-round approaches can be successfully combined with net-finding and quasi-uniform sampling offline algorithms to resolve open problems in the area. The first step in both the unweighted and weighted cases is to use an online algorithm for fractional hitting set, losing an $O(\log n)$-factor in the competitive ratio. Starting from [18], we know that rounding a fractional solution reduces to finding an ε-net in a set-system, where each element e with fractional value x_e is replaced by $\lceil nx_e \rceil$ *clones*. Since the fractional solution is feasible, each set in the original instance now contains at least n clones, so finding a hitting set in the original instance reduces to finding an ε-net (for $\varepsilon = 1/n$) that hits all sets of size at least n in this new set system $\mathcal{J}$. We start with the elegant algorithm of [22], which first samples each element with a certain base probability; now, for any large set T not hit by this sample, it selects elements of T independently with probability proportional to $1/|T|$ into the sample.

Carrying out this procedure in the online setting has two main issues: firstly, the fractional solution evolves over time, and the x_e values may increase as sets arrive. As a result, if an element e is contained in sets S_1 and S_2, the number of clones of e in the corresponding sets in the new set system $\mathcal{J}$ may be different, which affects the VC dimension and the SCC of $\mathcal{J}$. Secondly, the initial sampling step in [22] requires knowledge of all the elements in the set system; but in the online setting, new clones of an element appear when its fractional value increases. We handle the first issue by showing that the VC dimension and the SCC of the new set system $\mathcal{J}$ can be related to those of the original instance, and for the second issue, we replace the sampling step by a *deferred* sampling step.

The online algorithm for weighted hitting set adapts the quasi-uniform sampling algorithm of [8] to the online setting. This algorithm builds a chain of nested sets of clones, starting with the entire set V_0 of clones, with each subsequent set V_ℓ having size $({}^1/_2 + \delta_\ell)$ times the

size of $V_{\ell-1}$. Here δ_ℓ is a small bias term, which ensures that most sets S of the hitting set instance have at least half their elements survive to the next level. Sets whose size falls below $n/2^\ell$ are said to be "dangerous" and must select a "backup" element to hit them. The choice of this backup element is done to ensure that no element is picked too often. Apart from the challenge of the new clones arriving online, finding the backups seems like an inherently offline problem, since it seems to require knowing the entire set system. However, we view the problem of finding backup elements as a load-balancing problem, and show how to choose backups that increases the probability by a logarithmic factor. Existing presentations of the algorithm also build the layers iteratively, discarding sets that are covered by each layer to get the set system for the next layer – we observe that these optimizations are inessential, and removing them allows a clean online implementation. (See Algorithm 4 in Section 5.)

1.3 Related Work

Offline. Hitting set for geometric instances has been widely studied, see, e.g., [12, 7, 18, 11, 10, 26, 8]. The work of [7] gave an $O(d \log(d\,\mathsf{OPT}))$-approximation for the NP-hard hitting set problem in the offline case for set systems of VC dimension d. [18] reinterpreted their algorithm as first solving an LP relaxation for hitting set using multiplicative weights, and then rounding it using an ε-net algorithm. The use of shallow cell complexity in covering/hitting problems first appears in [11, 26], and was formally defined in [8]. These elegant papers gave the first results for *weighted* hitting set, where elements have costs, and the goal is to minimize the total cost of picked elements (as opposed to their cardinality); their approach is called *quasi-uniform* sampling. [1] extend the ε-net algorithm from [22] to the setting where the ε-net is in terms of element "sizes" and not their cardinality.

Online. [19] studied (unweighted) online hitting set in geometric settings, and gave optimal $\Theta(\log n)$-competitive ratio for hitting intervals on a line, and half-planes and unit disks in the plane $\mathbb{R}^2$. Their techniques did not extend to arbitrary disks; [15] gave an $O(\log n \log M)$-competitive algorithm for the setting where the disks have radii in $[1, M]$. Theorem 1 settles the question by giving an $O(\log n)$-competitive algorithm for arbitrary disks.

[21] give an $O(\log n)$-competitive algorithm for hitting set where the points are on the $n \times n$-grid, and the arriving sets are axis-aligned squares $S \subseteq [0, n)^2$. A partial answer to avoiding integrality of the points was given in [14] who give an $O(k^2 \log n \log M)$-competitive algorithm for homothetic copies of a regular k-gon for $k \geq 4$, with scaling factors in $[1, M]$. The above result of [15] extends this to positive homothets of any convex body in $\mathbb{R}^2$ with scaling factors in $[1, M]$. Theorem 1 settles the question for squares without any reliance on M; it remains unclear if our results can extend to positive homothets of convex bodies.

Our techniques do not seem to directly extend to the continuous setting (the "piercing set" problem), on which there is substantial work – see [9, 16, 17, 13]. Our techniques do not seem to give improved results for higher-dimensional covering/hitting problems (with the exception of hitting sets for half-spaces in $\mathbb{R}^3$), since the shallow-cell complexity $\varphi(\ell, k)$ of the associated set systems now depends polynomially on ℓ as well.

[5] give an $O(\log 1/\varepsilon)$-competitive algorithm to maintain *online ε-nets* for intervals and axis-aligned rectangles in the unweighted setting; since $\varepsilon = 1/\mathsf{OPT}$, and we lose an $O(\log n)$ factor in solving the fractional relaxation, their work can be combined with our framework to infer an $O(\log n (\log \mathsf{OPT})^2)$-competitive algorithm for hitting axis-aligned rectangles. In contrast, our general result for bounded VC dimension set systems gives an $O(\log n \log(\log n \cdot \mathsf{OPT}))$-competitive algorithm.

2 Notation and Definitions

Hitting Set: Offline and Online. $e \in U$ has a cost $c_e \geq 0$; the unweighted case is when $c_e = 1$ for all e. Let $n = |U|$ and $m = |\mathcal{F}|$. The LP relaxation for the *hitting set problem* is

$$\min_{x \in [0,1]^n} \Big\{ \sum_e c_e x_e \mid \sum_{e \in S} x_e \geq 1 \; \forall S \in \mathcal{F} \Big\}; \tag{1}$$

let $\mathsf{LP}(\mathcal{I})$ denote the optimal value of this LP on instance $\mathcal{I}$. Restricting the variables to be Boolean gives us the optimal set cover value, denoted $\mathsf{OPT}(\mathcal{I})$. Observe that $\mathsf{LP}(\mathcal{I}) \leq \mathsf{OPT}(\mathcal{I})$.

In an online fractional hitting set instance, an adversary fixes the instance $\mathcal{I}$, as well as an ordering $S_1, S_2, \ldots, S_m$ on the sets in $\mathcal{F}$; let $\mathcal{F}^t := \{S_1, \ldots, S_t\}$. The algorithm knows the element set U in advance. At each time $t \in [m]$, a set $S_t \in \mathcal{F}$ arrives and reveals identities of the elements in it. The algorithm must now produce a solution $\{x_e^t\}_{e \in U}$, such that x^t is a solution to the LP relaxation with instance $\mathcal{F}_t = (U, \mathcal{F}^t)$. The solution must be *monotone* – $x^t \leq x^{t+1}$ – in other words, elements cannot be dropped once they are picked. A (randomized) algorithm for this problem is α-*competitive* if $\mathbb{E}[\sum_{e \in U} c_e x_e^m] \leq \alpha \cdot \mathsf{OPT}(\mathcal{I})$, for every instance $\mathcal{I}$. If the solutions $x^t \in \{0,1\}^n$, then we have an algorithm for the online (integral) hitting set problem; we will denote the set of elements corresponding to x^t by C^t.

Dual Set Systems. The *dual set system* of the set system $\mathcal{I} = (U, \mathcal{F})$ is the set system $\mathcal{I}^* = (F, \mathcal{E})$, where we have a set E for every element e in U, an element s for every set $S \in \mathcal{F}$, and $s \in E \iff e \in S$. The *hitting set* problem for the system $\mathcal{I}$ is equivalent to the set cover problem on the dual system $\mathcal{I}^*$.

2.1 Shallow Cell Complexity

For a set system $(U, \mathcal{F})$ and a subset $U' \subset U$, denote the induced collection of sets as $\mathcal{F}_{U'} := \{S \cap U' \mid S \in \mathcal{F}\}$.

▶ **Definition 3** (Shallow Cell Complexity). *A set system $\mathcal{I} = (U, \mathcal{F})$ is said to have* shallow cell complexity *(SCC) $\varphi : \mathbb{Z}_+ \times \mathbb{Z}_+ \to \mathbb{Z}_+$ if for any $\ell, k \in \mathbb{Z}_+$, and for any subset $U' \subset U$ with $|U'| = \ell$, the number of distinct subsets of size at most k in $\mathcal{F}_{U'}$ is at most $\ell \cdot \varphi(\ell, k)$.*

Crucially, the number of distinct sets in $\mathcal{F}_{U'}$ is bounded by $\ell \cdot \varphi(\ell, k)$ rather than the worst-case $\binom{\ell}{k}$. If the VC dimension of the set system $\mathcal{I}$ is d, the shattering lemma [25, Ch. 15] implies that $\ell \cdot \varphi(\ell, k) \leq \binom{\ell}{d} \approx (\mathrm{e}\ell/d)^d$.

We focus on functions φ which are well-behaved, in the sense of [23].

▶ **Definition 4** (Well-Behaved SCC). *A function $\varphi(\cdot, \cdot)$ is (a, b)-*well-behaved *for constants $a \in (1, 2)$ and $b \geq 2$ if it is non-decreasing in both arguments, and for all positive integers $\ell \geq k \geq b$, we have $\varphi(\ell, k) \leq (\varphi(\ell/2, k/2))^a$.*

Well-behavedness is easy to verify for the common case of $\varphi(\ell, k) = k^c$ for some constant $c \geq 0$. Henceforth, our focus will be on such functions.

3 Uniform Set Systems

The first step of our approach is the following "uniformization" procedure, that takes any online algorithm $\mathcal{A}$ for (fractional) hitting set, and some instance $\mathcal{I} = (U, \mathcal{F})$ with n elements and m sets, and produces a new instance $\mathcal{J} = (V, \mathcal{G})$ with $N := n(n+1)$ elements and

$M := m$ sets in an online manner. For each set $S_j \in \mathcal{F}$, we have a corresponding set $T_j \in \mathcal{G}$. For each element $e \in U$, there are at most $n+1$ "clone" elements $\{(e,i) \mid i \in \{0,\dots,n\}\}$ in V, each of the same cost as that of e.

Recall that the sets in $\mathcal{F}$ appear in the order $S_1,\dots,S_m$. The algorithm $\mathcal{A}$ maintains a fractional solution x^t at time t (i.e., after the arrival of the set S_t). Without loss of generality, we assume that the fractional values x^t_e never exceed 1 – indeed, if these values ever exceed 1, we can reduce them to 1 without violating feasibility. The element-set containment is defined in Algorithm 1. When the set S_t arrives, the algorithm $\mathcal{A}$ updates the fractional solution to x^t. The solution x^t is now used to define the elements in the corresponding set T_t in the instance $\mathcal{J}$: $T_t := \{(e,i) : e \in S_t, i \le \lceil n \cdot x^t_e \rceil\}$.

Before analyzing this algorithm, we give a useful definition:

▶ **Definition 5** (Projection). *For a subset $T \subseteq V$, define its projection, $\mathsf{proj}(T) := \{e : (e,i) \in T \text{ for some } i\}$, as the set of elements in U whose clones are present in T.*

Algorithm 1 Uniformize($\mathcal{I}, \mathcal{A}$).

```
for sets S_1 to S_m do
    pass set S_t to algorithm A, and let x^t be the resulting solution
    define a new set T_t ∈ G as {(e,i) | e ∈ S_t, i ≤ ⌈n · x^t_e⌉}.
```

▶ **Theorem 6.** *Suppose $\mathcal{A}$ is a (deterministic) ρ-competitive online algorithm for fractional set cover. Given an instance $\mathcal{I} = (U,\mathcal{F})$, let Uniformize($\mathcal{I},\mathcal{A}$) generate the instance $\mathcal{J} = (V,\mathcal{G})$. Then the instance $\mathcal{J}$ satisfies the following properties:*

- **(i)** *Any solution for $\mathcal{I}$ can be converted into a solution for $\mathcal{J}$ of the same cost, and vice versa.*
- **(ii)** *Each set $T_t \in \mathcal{G}$ contains at least $B := n$ elements in $\mathcal{G}$.*
- **(iii)** *$N' \le B(1 + \sum_e x_e)$, where N' denotes the number of elements in V contained in at least one set in $\mathcal{G}$. Hence, in the unweighted setting, $N' \le B(1 + \rho\,\mathsf{OPT}(\mathcal{I}))$.*
- **(iv)** *We have $\mathsf{VCdim}(\mathcal{G}) \le \mathsf{VCdim}(\mathcal{F})$, also $\varphi_{\mathcal{G}}(\ell,k) \le k \cdot \varphi_{\mathcal{F}}(\ell,k)$.*

Proof. For the first claim, let $U' \subseteq U$ be a solution to $\mathcal{I}$. Then, $V' := \{(e,0) : e \in U'\}$ is a solution to $\mathcal{J}$. Conversely, if V' is a solution to $\mathcal{J}$, then $\mathsf{proj}(V')$ is a solution to $\mathcal{I}$. The second claim follows from the feasibility of x^t. Indeed, for each set S_t, $\sum_{e \in S_t} \min(x^t_e, 1) \ge 1$. Therefore, $\sum_{e \in S_t} \lceil n \min(x^t_e, 1) \rceil \ge n$. But the left-hand side above is exactly equal to $|T_t|$.

To prove property iii, observe that the number of clones of an element e is at most $nx^m_e + 1$. Summing over all elements, we see that $N' \le n(\sum_e x_e + 1)$. In the unweighted setting, $\sum_e x_e \le \rho \cdot \mathsf{OPT}(\mathcal{I})$.

For property iv, we first consider the VC dimension of $\mathcal{G}$. Let $V' \subseteq V$ be shattered by the sets in $\mathcal{G}$. We first observe that V' contains at most one clone corresponding to an element $e \in U$. Indeed, suppose V' contains (e,i) and (e,i') for two distinct indices $i < i'$. Now every set $T \in \mathcal{G}$ containing (e,i') also contains (e,i). Thus, there is no set in $\mathcal{G}$ that contains (e,i'), but does not contain (e,i), a contradiction. This implies that the corresponding set $U' = \{e : (e,i) \in V' \text{ for some } i\}$ is also shattered by $\mathcal{F}$. Thus, $\mathsf{VCdim}(\mathcal{G}) \le \mathsf{VCdim}(\mathcal{F})$.

Finally, we bound the SCC of the set system $\mathcal{J}$. Fix two positive integers ℓ and k. Let V' be a subset of V with $|V'| \le \ell$. Let $\mathcal{G}'$ be the distinct sets in $\mathcal{G}_{V'}$ of size at most k. Let U' denote $\mathsf{proj}(V')$ and $\mathcal{F}'$ denote $\{\mathsf{proj}(T) : T \in \mathcal{G}'\}$. Since $|U'| \le \ell$ and each set in $\mathcal{F}'$ has size at most k, $|\mathcal{F}'| \le \ell \cdot \varphi_{\mathcal{F}}(\ell,k)$.

Consider a set $S \in \mathcal{F}'$ and let $\mathcal{G}'(S)$ be the collection of sets $T \in \mathcal{G}'$ for which $\mathsf{proj}(T) = S$. We claim that if $T, T' \in \mathcal{G}'(S)$ then either $T \subseteq T'$ or $T' \subseteq T$. Indeed, suppose $T = T_t \cap V'$ and $T' = T_{t'} \cap V'$ for some times t and t' respectively. Assume w.l.o.g. that $t < t'$. Then $x_e^t \leq x_e^{t'}$ for each element e. Therefore, if $(e, i) \in T$, then $e \in S$. Therefore $(e, i) \in T'$ as well. This shows that $T \subseteq T'$. Since each of the sets in $\mathcal{G}'(S)$ has at most k elements, we see that $|\mathcal{G}'(S)| \leq k$. Summing over all the sets $S \in \mathcal{F}'$, we see that $|\mathcal{G}'| \leq \ell \cdot \varphi_{\mathcal{F}}(\ell, k) \cdot k$. This shows that $\varphi_{\mathcal{G}}(\ell, k) \leq k \cdot \varphi_{\mathcal{F}}(\ell, k)$. ◀

Since we can perform this "uniformization" process to go from $\mathcal{I}$ to $\mathcal{J}$ in an online fashion – and we can also convert solutions to $\mathcal{J}$ back to solutions to $\mathcal{I}$ online – we henceforth imagine working with the instance $\mathcal{J}$, and want to hit every set of $\mathcal{J}$ by picking one of the B elements in it.

4 Unweighted Online Hitting Set

In this section, we study the unweighted version of the online hitting set problem. Consider the instance $\mathcal{J} = (V, \mathcal{G})$ generated by the Uniformize routine in Algorithm 1. We rely on the structural properties established in Theorem 6: there exists a subset $V' \subseteq V$ of size $|V'| = N'$ with the guarantee that each element in V' is contained in at least one set in $\mathcal{G}$. Moreover, each set in $\mathcal{G}$ contains at least B elements, and our solution must hit this set. We first describe the offline algorithm of [23], and then extend it to an online algorithm.

4.1 The NetFinder Algorithm

We consider the following offline algorithm for finding ε-nets (see Algorithm 2) [23]. The algorithm maintains a subset H of V', which is initialized to the empty set. In the base sampling step (line 2), each element in V' is added to H with probability α/B. In the alteration step (line 5), we consider each set T_t. While a set T_t is not yet hit by some element in H, we add each element in T_t to the set H independently with probability $\beta/|T_t|$.

Algorithm 2 Net-Finder-V1.

```
1 for each element e ∈ V' do
2 |  select e into H independently w.p. α/B ;          // Base Sampling
3 for sets T_1 to T_m do
4 |  while T_t is not hit by H do
5 |  |  add each element in T_t into H independently w.p. β/|T_t| ;   // Alteration
```

Let ε denote B/N'. Since each set in $\mathcal{G}$ has size at least B, any ε-net in the set system $(V', \mathcal{G})$ must hit all the sets in $\mathcal{G}$. The following guarantee now follows from [23, Theorem 7.6]:

▶ **Theorem 7.** *Assume that the set system $\mathcal{G}$ has SCC $\varphi_{\mathcal{G}}$ and VC dimension d. Define $\varepsilon = B/N'$. For $\alpha = O(d + \ln(d \cdot \varphi_{\mathcal{G}}(O(d/\varepsilon), O(d)))) \cdot O(1/\varepsilon)$ and $\beta = O(d)$, the Net-Finder-V1 Algorithm picks $O(1/\varepsilon) \cdot \log \varphi_{\mathcal{G}}\big(O(d/\varepsilon), O(d)\big) + O(d/\varepsilon)$ elements in expectation.*

4.2 Implementing NetFinder Online

We show how to implement Algorithm 2 online. There are two technical difficulties:

(i) The parameter α used in Algorithm 2 depends on N' (see Theorem 7). But in the online setting, elements contained in a set are revealed over time, and hence N' changes over time. We address this using a standard doubling trick: we divide the execution into *phases*. In a single phase, the value of N' lies in the range $[2^a, 2^{a+1})$ for some non-negative integer a.

(ii) The base sampling step in Algorithm 2 should not be performed for elements $e \in V \setminus V'$. As observed above, the set V' only gets revealed over time. Crucially, base sampling can be deferred: we sample an element $e \in V'$ only upon the arrival of the first set T_t containing it.

The algorithm, Net-Finder-V2, is given in Algorithm 3. First, we find the nearest power of 2 which is larger than N' and define ε accordingly (line 3). Observe that the quantity α is a function of ε (as given in Theorem 7). The rest of the algorithm is essentially the same as Net-Finder-V1, but we perform base sampling on an element only when the first set containing it arrives. Thus, we classify the elements in V as being either *fresh* or *stale*; initially, each element is fresh. An element becomes stale when we encounter the first set containing it. The alteration steps are then done exactly as in Net-Finder-V1. It follows that the statement of Theorem 7 holds for this algorithm as well.

Algorithm 3 Net-Finder-V2.

```
1 for sets T_1 to T_m do
2     Update V' := ∪_{t'≤t} T_{t'} and N' = |V'|
3     Let a be such that N' ∈ [2^a, 2^{a+1}).  Set ε = B/2^{a+1}.
4     for each fresh element e ∈ T_t do
5         select e into H independently with probability α(ε)/B ;   // Base Sampling
6         mark e as stale
7     while T_t is not hit by H do
8         select each element in T_t independently w.p. β/|T_t| ;   // Alteration
```

4.3 The Final Algorithm

We now describe the online hitting set algorithm combining Algorithm 1 and Algorithm 3. If $\mathcal{A}$ is an online algorithm for fractional hitting set and $\mathcal{I} = (U, \mathcal{F})$ an instance of the online hitting set, we use Algorithm 1 to obtain the instance $\mathcal{J} = (V, \mathcal{G})$, such that each set $S_t \in \mathcal{F}$ gives rise to set $T_t \in \mathcal{G}$. In each iteration t, Net-Finder-V2 maintains H as a valid hitting set for $\mathcal{J}$ by potentially adding elements from T_t. Then, we recover the corresponding hitting set for $\mathcal{I}$ by projecting the solution back, i.e., using the set $\mathsf{proj}(H)$. We get the following result:

▶ **Theorem 8.** *Given an online ρ-competitive algorithm for fractional online hitting set, there is an online randomized algorithm for unweighted hitting set that has expected competitive ratio*

$$O(\rho\, d) + O\left(\rho \log\left(d\, \varphi_{\mathcal{F}}(\rho\, d\, \mathsf{OPT}(\mathcal{I}), d)\right)\right)$$

for any instance $\mathcal{I} = (V, \mathcal{F})$, where the set system $\mathcal{F}$ has SCC $\varphi_{\mathcal{F}}$ and VC dimension d.

Proof. We need to bound the expected size of $\mathsf{proj}(H)$. Since $|\mathsf{proj}(H)| \le |H|$, it suffices to bound $\mathbb{E}[|H|]$. Consider a phase during the algorithm when N' stays in the range $[2^a, 2^{a+1})$ for some a. Observe that $\varepsilon = B/2^{a+1}$.

Using the notation in Section 4.1, Theorem 7 implies that $\mathbb{E}[|H|]$ is at most

$$O(1/\varepsilon)\cdot\log\varphi_{\mathcal{G}}\big(O(d/\varepsilon), O(d)\big)+O(d/\varepsilon) = O(2^{a+1}/B)\cdot\log\varphi_{\mathcal{G}}\big(O(d2^{a+1}/B), O(d)\big)+O(d2^{a+1}/B).$$

Summing over all a (which forms a series dominated by the last term) and using N' to denote the final value of $|V'|$, we see that the $\mathbb{E}[|H|]$ is at most

$$O(N'/B)\cdot\log\varphi_{\mathcal{G}}\big(O(N'/B), O(d)\big) + O(dN'/B).$$

Theorem 6 gives $N'/B \le 1 + \rho\,\mathsf{OPT}(\mathcal{I})$ and $\varphi_{\mathcal{G}}(\ell, k) \le k\varphi_{\mathcal{F}}(\ell, k)$, and hence the proof. ◀

▶ **Corollary 9.** *Given an online ρ-competitive algorithm for fractional online hitting set, there is an online randomized algorithm for (unweighted) online hitting set that has expected competitive ratio $O(d\rho \cdot \log(d\rho \cdot \mathsf{OPT}(\mathcal{I})))$, if the input set system $\mathcal{F}$ has VC dimension d.*

Furthermore, if the SCC $\varphi_{\mathcal{F}}$ of the set system satisfies $\varphi_{\mathcal{F}}(\ell, k) = \mathrm{poly}(k)$, then the expected competitive ratio is $O(d\rho \log d)$.

Proof. If the set system has VC dimension d, then $\varphi_{\mathcal{F}}(\ell, k) = O(\ell^d)$. Theorem 8 now implies the desired bound on the competitive ratio. The result for the case $\varphi_{\mathcal{F}}(\ell, k) = \mathrm{poly}(k)$ also follows from Theorem 8. ◀

Setting $\rho = \log n$ in Corollary 9 implies Theorem 1.

5 Online Algorithm for Weighted Hitting Sets

We now consider the weighted case of the hitting set problem on an online instance $\mathcal{J} = (V, \mathcal{G})$. We use $N = |V|$ to denote the size of the universe, and $B = n$ to denote a lower bound on the size of each set in this instance. In lieu of the size guarantee, we give a per-element guarantee and show that each element $e \in V$ which lies in at least one set is chosen with probability at most α/B, for some parameter $\alpha = \alpha(N, B) \geq 1$. We adapt the quasi-uniform sampling framework of [8] to the online setting. For convenience, we follow the exposition of that algorithm as presented in [23, Chapter 8].

5.1 Element Levels

The idea of the algorithm is natural: we construct a nested family of sets starting with $V_0 = V$, where each subsequent set $V_{\ell+1}$ is a uniformly random subset of size roughly $(1/2 + \delta)$ times that of the previous set V_ℓ, for a small "bias" term δ. The process ends at a level L^* with the corresponding set V_{L^*} having size $\approx N/B$. If we chose V_{L^*} as our hitting set H, the probability of each element $e \in H$ would be $\approx 1/B$, as desired. As sets in $\mathcal{G}$ are hit only with constant probability by V_{L^*}, we require "alterations" (backups) to ensure validity.

We assume that $\varphi_{\mathcal{G}}$ that is (a, b)-well-behaved for some positive constants a, b. (Recall Definition 4.) Let c_1 be a large enough constant and assume $B \geq c_1$. For brevity, define

$$N_\ell = \frac{c_1 N}{2^\ell} \qquad B_\ell = \frac{B}{2^\ell}. \tag{2}$$

Now we can define the element levels as follows:

1. Define L^* to be the largest value $\ell \in [0, \log B/c_1]$ such that

$$h_{\mathcal{G}}(N_\ell, B_\ell) \overset{\text{(def.)}}{=} \sqrt{\frac{8 \ln(B_\ell \cdot \lceil \log N_\ell \rceil \cdot \varphi_{\mathcal{G}}(N_\ell, B_\ell))}{B_\ell}} \leq 1/2. \tag{3}$$

   If $L^* = 0$ (or if $h_{\mathcal{G}}(N_0, B_0) > 1/2$), we return V as the hitting set. Since we have $L^* = \log \frac{B}{c_1}$ or $h_{\mathcal{G}}(N_{L^*}, B_{L^*}) > 1/2$, it follows that

$$\frac{1}{2^{L^*}} = O\left(\frac{1}{B} + \frac{\log(\log N \cdot B_{L^*} \cdot \varphi_{\mathcal{G}}(N_{L^*}, B_{L^*}))}{B}\right) = O\left(\frac{\log(\log N \cdot B_{L^*} \cdot \varphi_{\mathcal{G}}(N_{L^*}, B_{L^*}))}{B}\right). \tag{4}$$

2. Define $V_0 = V$. For each $\ell \in \{0, 1, \ldots, L^* - 1\}$, let $V_{\ell+1}$ be a random subset of V_ℓ of size

$$|V_\ell| \cdot (1/2 + h_{\mathcal{G}}(|V_\ell|, B_\ell)).$$

3. The *level* of element e is defined as the largest $\ell \in \{0, 1, \ldots, L^*\}$ such that $e \in V_\ell$.

Since the elements of V are given up-front, we define subsets V_ℓ before sets in $\mathcal{G}$ arrive.

5.2 The Weighted Hitting Set Algorithm

Given the notion of element levels, the main idea of the algorithm of [8], is the following: for each set $T \in \mathcal{G}$, consider the first level where $|T \cap V_\ell|$ is smaller than $B/2^\ell$. If this (low-probability) bad event happens, we pick some element in T at level $\ell - 1$ to hit it (since $|T \cap V_0| = |T| \geq B$, $\ell > 0$). The choice of this "backup" element is done consistently so that no element is chosen with too high a probability – showing such backup elements can be chosen is precisely where the small SCC of set system plays a role.

Algorithm 4 Offline Small SCC Hitting Set.

1 Define element levels and sets $V_\ell \leftarrow \{e \in V \mid \mathsf{level}(e) \geq \ell\}$ as in Section 5.1
2 **foreach** *arriving set* $T \in \mathcal{G}$ **do**
3 **foreach** *level* $\ell \in \{0, 1, 2, \ldots, L^* - 1\}$ **do**
4 define backup element $\gamma_\ell(T) \in V_\ell \cap T$ as in Section 5.3.1
5 **if** $|T \cap V_{\ell+1}| < B/2^{\ell+1}$ **then**
6 add element $\gamma_\ell(T)$ to the hitting set Q_ℓ
7 Let V' be the elements in V contained in at least one set in $\mathcal{G}$
8 **return** $H = Q_0 \cup \ldots \cup Q_{L^*-1} \cup (V_{L^*} \cap V')$

The algorithm (presented in 4) follows this outline: For each arriving set T of size at least B, and each level ℓ, we define a backup element, namely $\gamma_\ell(T)$, that will hit it in case of the bad event happening at the next level. This bad event at level ℓ is that the size of $T \cap V_\ell$ is smaller than $B/2^\ell$. Since each arriving set T is of size at least B, it either experiences a bad event at some level $\{0, 1, \ldots, L^* - 1\}$, or intersects the final set V_{L^*}. It is now easy to check that the solution returned by the algorithm is feasible; the remaining part is to show that each element belongs to this hitting set with probability α/B for some suitable parameter α.

In order to implement this algorithm in an online fashion, we need to define the backup element in line 4 in an online fashion; we do this in the next section with only a small loss in performance over the offline setting.

5.3 Backup Elements and Consistent Rounding

The next lemma defines a good backup map that can be maintained online:

▶ **Lemma 10.** *Consider a set system $(X, \mathcal{H})$. For any positive integers k and i, let $\mathcal{H}_i$ be the subsets in $\mathcal{H}$ of sizes in $[2^i k, 2^{i+1}k)$. Then there exists a map $\gamma^* : \mathcal{H} \to X$ such that $\gamma^*(S) \in S$ for each $S \in \mathcal{H}$ and for each integer $i \geq 0$, the number of sets from the subcollection $\mathcal{H}_i$ (i.e., the "scale-i" sets) mapped by γ^* to any element $e \in X$ is at most*

$$\mathrm{load}^*_{\mathcal{H}}(k, i) := \varphi_{\mathcal{H}}(|X|, 2^{i+1}k) \cdot 2^{i+1}k.$$

Moreover, if the sets in $\mathcal{H}$ arrive online, we can maintain such a map γ such that the number of sets assigned to each element is at most $\mathrm{load}_{\mathcal{H}}(k, i) := \lceil \log |X| \rceil \cdot \mathrm{load}^*_{\mathcal{H}}(k, i)$.

Proof. The first part of the lemma is proved in [23, Lemma 8.18]. For the second claim, we do the following. Fix an integer $i \geq 0$. We build a map $\gamma_i : \mathcal{H}_i \to X$ online using a basic result about algorithms for online load-balancing: it says that if jobs arrive online and each job must be assigned to one of some subset of "valid" machines, then the greedy algorithm ensures that the number of jobs assigned to any machine is at most $\lceil \log \#\text{machines} \rceil$ factor higher than the optimal maximum load [3]. In our setting, the jobs are sets from $\mathcal{H}_i$ and the

machines are elements of X, which shows that the load of each element with respect to the sets in $\mathcal{H}_i$ is only $\lceil \log N \rceil$ worse than in γ^*. Finally, the map $\gamma : \mathcal{H} \to X$ merely combines all these maps $\{\gamma_i\}$ into one, completing the claim. ◀

The following "bias" term will be crucial for the rest of the analysis:

$$h_{\mathcal{H}}(\ell, k) := \sqrt{\frac{8 \ln(k \cdot \lceil \log \ell \rceil \cdot \varphi_{\mathcal{H}}(\ell, k))}{k}}. \tag{5}$$

▶ **Lemma 11.** *Consider set system $(X, \mathcal{H})$ and a positive integer k such that $|S| \geq k$ for all $S \in \mathcal{H}$, and where $h_{\mathcal{H}}(|X|, k) \leq 1/2$. Let $Y \subseteq X$ be a uniformly random subset of size $|X| \cdot \big(1/2 + h_{\mathcal{H}}(|X|, k)\big)$. Suppose the sets in $\mathcal{H}$ are revealed online one by one. Let γ be the online mapping defined in Lemma 10. Then, defining*

$$Q := \{\gamma(S) \mid S \in \mathcal{H}, |S \cap Y| < k/2\}$$

gives us a set $Q \subseteq X$ (which is maintained in an online fashion) such that

- **(i)** *for each $S \in \mathcal{H}$, either $|S \cap Y| \geq k/2$ or $S \cap Q \neq \varnothing$, and*
- **(ii)** *each $e \in X$ belongs to Q w.p. $O(1/k^2)$, where the probability is over the choice of Y.*

Proof. This proof follows that of [23, Lemma 8.13], the only differences are that (a) the procedure is performed online, and (b) the load of any element is higher by $\lceil \log |X| \rceil$ due to the online load balancing in Lemma 10. Accordingly, the second term $h_{\mathcal{H}}(|X|, k)$ in the sampling probability contains a term corresponding to $\lceil \log |X| \rceil$, which helps to control the probability of any element being chosen into Q.

We now give a sketch of the proof. A Chernoff bound implies that if we pick a random subset $Y \subseteq X$ of size $|X| \cdot (1/2 + \delta)$ with $\delta \leq 1/2$, any set S of size at least k has

$$\Pr[|S \cap Y| < k/2] \leq \exp(-|S|/2 \cdot \delta^2). \tag{6}$$

(See [6] for a proof.) As in the statement of Lemma 10, we partition $\mathcal{H}$ into sets $\mathcal{H}_i$ based on the set sizes. Now we add an element e to Q if $e = \gamma(S)$ for some set $S \in \mathcal{H}$ and $|S \cap Y| \leq k/2$. By a union bound, the probability of e being added to Q is at most

$$\Pr[e \in Q] \leq \sum_{i \geq 0} \sum_{S \in \mathcal{H}_i : \gamma(S) = e} \Pr[|S \cap Y| < k/2] \leq \sum_{i \geq 0} \mathrm{load}_{\mathcal{H}}(k, i) \cdot \exp(-2^{i-1} k \cdot h_{\mathcal{H}}(|X|, k)^2).$$

For brevity, define $\Lambda := \lceil \log |X| \rceil$. Using the definition of $h_{\mathcal{H}}(\ell, k)$ in (5) and Lemma 10, we see that the above expression is at most

$$\leq \sum_{i \geq 0} \frac{2^{i+1} k}{k^{2^{i+2}}} \frac{\Lambda \cdot \varphi_{\mathcal{H}}(|X|, 2^{i+1} k)}{(\Lambda \cdot \varphi_{\mathcal{H}}(|X|, k))^{2^{i+2}}} \overset{(\star)}{\leq} \sum_{i \geq 0} \frac{2^{i+1} k}{k^{2^{i+2}}} \underbrace{\frac{(\Lambda \cdot \varphi_{\mathcal{H}}(|X|, k))^{2(i+1)}}{(\Lambda \cdot \varphi_{\mathcal{H}}(|X|, k))^{2^{i+2}}}}_{\leq 1} \leq O(1/k^2),$$

where inequality $(\star)$ used the fact that $\Lambda = \lceil \log |X| \rceil \geq 1$, $\varphi_{\mathcal{H}}$ is (a, b)-well-behaved and that $k \geq b$ (we assume the constant in $O(1/k^2)$ is at least b^2 and hence the upper bound on the probability of this event is trivial if $k < b$). This completes the proof of the lemma. ◀

In summary, we know that for any set system $(X, \mathcal{H})$, if we pick a random subset Y of about half (plus some "bias" term $h_{\mathcal{H}}(|X|, k)$ that depends on the SCC and the extra load $\lceil \log |X| \rceil$) of the elements, then most sets of size at least B have at least $B/2$ elements in common with Y, and moreover, each element $e \in X$ is picked as a backup element (due to sets that have fewer than $B/2$ elements) with probability at most $O(1/k^2)$.

5.3.1 Defining backup maps in Algorithm 4

We can now use Lemma 10 to define the maps γ_ℓ used in line 4 of Algorithm 4. Fix an index $\ell \in \{0, \ldots, L^* - 1\}$. Consider the set system $(V_\ell, \mathcal{G}_\ell)$ where $\mathcal{G}_\ell$ is the collection of subsets $T \in \mathcal{G}$ for which $|T \cap V_\ell| \geq B/2^\ell$. We now use Lemma 10 with $X = V_\ell, \mathcal{H} = \mathcal{G}_\ell$ and $k = B_\ell = B/2^\ell$. We set γ_ℓ to be the resulting online map $\gamma : \mathcal{G}_\ell \to V_\ell$. Observe that the set Q defined in Lemma 10 is same as the set Q_ℓ defined in Algorithm 4. We show next that maps γ_ℓ (and the resulting backup sets Q_ℓ) ensure a low probability of picking elements.

5.4 Probability of Picking Elements

To bound the probability of selecting an element e in the hitting set constructed by Algorithm 4, we first relate $h_{\mathcal{G}}\big(N_\ell, B_\ell\big)$ for varying values of ℓ (See [6] for the proofs).

▷ Claim 12. For any $\ell < L^*$ we have $h_{\mathcal{G}}\big(N_{\ell+1}, B_{\ell+1}\big) \geq \sqrt{\frac{2}{a}} \cdot h_{\mathcal{G}}\big(N_\ell, B_\ell\big)$.

▷ Claim 13. For any $\ell \leq L^*$ we have $|V_\ell| \leq c_1\, N/2^\ell$, and for each $e \in V$, $Pr[e \in V_\ell] \leq \frac{c_1}{2^\ell}$.

▶ **Theorem 14.** *In Algorithm 4, each element $e \in V$ is selected in the set H with probability at most $O\big(\frac{1}{B} \cdot \log(B_{L^*} \cdot \log N \cdot \varphi_{\mathcal{G}}(N_{L^*}, B_{L^*}))\big)$.*

Proof. Recall that if $L^* = 0$, we return the entire set V as the solution (which clearly hits all sets of size at least B); using (4), we get

$$\Pr[e \in H] = 1 \leq O\bigg(\frac{\log(B_L \cdot \log N \cdot \varphi_{\mathcal{G}}(c_1 N, N))}{B}\bigg). \tag{7}$$

Else, by a union bound, the probability that $e \in H$ is

$$\sum_{\ell=0}^{L^*-1} \Big(\Pr[e \in V_\ell] \Pr[e \in Q_\ell \mid e \in V_\ell]\Big) + \Pr[e \in V_L^*] = \sum_{\ell=0}^{L^*-1} \frac{c_1}{2^\ell} \cdot (B_\ell)^{-2} + \frac{c_1}{2^{L^*}},$$

where we used Claim 13 and Lemma 11ii for the first expressions, and Claim 13 again for the second. Now simplifying using the definition (2) for the first term, and the inequality (4) for the second, we get

$$\Pr[e \in H] \leq \frac{c_1}{B^2} \cdot \sum_{\ell=0}^{L^*-1} 2^\ell + O\bigg(\frac{\log(\log N \cdot B_{L^*} \cdot \varphi_{\mathcal{G}}(N_{L^*}, B_{L^*}))}{B}\bigg).$$

Finally, the geometric sum gives $2^{L^*} \leq \frac{B}{c_1}$, and results in the claimed bounds. ◀

5.5 The Algorithm for Weighted Hitting Sets

We now combine Algorithm 1 and Algorithm 4 to get an online weighted hitting set algorithm. We first give a simple observation about Algorithm 1 (See [6] for the proof).

▷ Claim 15. Let $\mathcal{I} = (U, \mathcal{F})$ be an instance of the weighted online hitting set problem. Then, Algorithm 1 can be implemented such that (let x_e be the solution at the end of the instance) (i) $\sum_e c_e x_e \leq 4\rho\, \mathsf{OPT}(\mathcal{I})$, (ii) if $x_e^t \neq 0$ for any time t and element e, then $x_e^t \geq 1/n$.

The online weighted hitting set algorithm is as follows: Algorithm 1 (with the changes in Claim 15) takes an instance $\mathcal{I} = (U, \mathcal{F})$ of the weighted hitting set problem and outputs an online instance $\mathcal{J} = (V, \mathcal{G})$ (which has the properties stated in Theorem 6). Since the backup maps γ_ℓ can be defined in an online manner, Algorithm 4 can be implemented on the instance $\mathcal{J}$ in an online manner. This gives us an online hitting set H. The hitting set for $\mathcal{I}$ is just $\mathsf{proj}(H)$. We now state the main result about the competitive ratio of this algorithm.

▶ **Theorem 16.** *Given an online ρ-competitive algorithm for fractional online hitting set, there is a randomized (weighted) online hitting set algorithm with expected competitive ratio*

$$O\big(\rho \cdot \log(B_{L^*} \log n \cdot \varphi_{\mathcal{F}}(N_{L^*}, B_{L^*}))\big).$$

Here L^* is the level associated with the instance $(V, \mathcal{G})$ as given by (3)*.*

Proof. Let e be an element in V. If e does not appear in any of the sets $\cup_t T_t$, then it is not selected by Algorithm 4. Otherwise its selection probability is upper bounded by Theorem 14.

Consider an element $e \in U$ (note that U is the ground set of the original instance $\mathcal{I}$). We now bound the probability that it gets selected. Let x_e be the final fractional value of e in Algorithm 1. If $x_e = 0$, then no clones of e appear in any of the sets in the instance $\mathcal{G}$, and hence, none of these clones gets added to H in Algorithm 4. Therefore probability of e getting selected is 0. Now suppose $x_e > 0$. Claim 15 shows that $x_e \geq 1/n$. Let Y_e be the set of clones corresponding to e in the set V. In other words, $Y_e = \{(e, i) : i \leq \lceil \min(1, nx_e) \rceil\}$. Since $x_e \geq 1/n$, $|Y_e| \leq 2nx_e$. Now, any clone (e, i) gets added to H with probability at most

$$O\Big(\frac{1}{B} \cdot \log(B_{L^*} \log N \cdot \varphi_{\mathcal{G}}(N_{L^*}, B_{L^*}))\Big).$$

Therefore, the probability that any of the clones of e gets added to H is at most $|Y_e|$ times the above expression. Thus, the probability of selection of e is at most (recall that $B = n$)

$$O\big(\log(B_{L^*} \log N \cdot \varphi_{\mathcal{G}}(N_{L^*}, B_{L^*}))\big) x_e.$$

The desired result now follows from the fact that $\sum_e c_e x_e = 4\rho \cdot \mathsf{OPT}$ (using Claim 15), that $N = O(n^2)$ and that $\varphi_{\mathcal{G}}(N_{L^*}, B_{L^*}) \leq B_{L^*} \cdot \varphi_{\mathcal{F}}(N_{L^*}, B_{L^*})$ due to Theorem 6. ◀

The guarantee of Theorem 16 depends on the value of L^*, which depends on the form of φ. We now apply the above result for the most interesting case: that of set systems with linear SCC, i.e., where $\varphi_{\mathcal{F}}(\ell, k) = \text{poly}(k)$.

▶ **Corollary 17** (Linear SCC)**.** *Suppose we are given an online ρ-competitive algorithm for fractional online weighted hitting set and the arriving instance $\mathcal{I} = (U, \mathcal{F})$, $n := |U|$, satisfies $\varphi_{\mathcal{F}}(\ell, k) = \text{poly}(k)$. Then there is an online randomized algorithm for (weighted) online hitting set that has expected competitive ratio $O\big(\rho \log \log n\big)$.*

Corollary 17, whose proof is deferred to the full version of the paper, implies Theorem 2.

References

1. Sander Aarts and David B. Shmoys. Hitting sets when the shallow cell complexity is small. In Jaroslaw Byrka and Andreas Wiese, editors, *WAOA*, volume 14297 of *Lecture Notes in Computer Science*, pages 160–174. Springer, 2023. doi:10.1007/978-3-031-49815-2_12.
2. Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009. doi:10.1137/060661946.
3. Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line assignments. *J. Algorithms*, 18(2):221–237, 1995. doi:10.1006/JAGM.1995.1008.
4. Nikhil Bansal and Kirk Pruhs. The geometry of scheduling. *SIAM Journal on Computing*, 43(5):1684–1698, 2014. doi:10.1137/130911317.
5. Sujoy Bhore, Devdan Dey, and Satyam Singh. Online epsilon net and piercing set for geometric concepts. In *ICLR 2025*, 2025. URL: https://openreview.net/forum?id=nNiWRRj6r9.
6. Sujoy Bhore, Anupam Gupta, and Amit Kumar. Improved online hitting set algorithms for structured and geometric set systems. *Preprint*, abs/2603.14293, 2026. arXiv:2603.14293.

7 Hervé Brönnimann and Michael T Goodrich. Almost optimal set covers in finite VC-dimension. In *SoCG*, pages 293–302, 1994. doi:10.1145/177424.178029.

8 Timothy M. Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *SODA*, pages 1576–1585. SIAM, 2012. doi:10.1137/1.9781611973099.125.

9 Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004. doi:10.1137/S0097539702418498.

10 Chandra Chekuri, Kenneth L Clarkson, and Sariel Har-Peled. On the set multicover problem in geometric settings. *ACM Transactions on Algorithms (TALG)*, 9(1):1–17, 2012. doi:10.1145/2390176.2390185.

11 Kenneth L. Clarkson and Kasturi R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discret. Comput. Geom.*, 37(1):43–58, 2007. doi:10.1007/S00454-006-1273-8.

12 KL Clarkson and Peter W Shor. Applications of random sampling in computational geometry, ii. *Discrete & computational geometry*, 4(5):387–422, 1989. doi:10.1007/BF02187740.

13 Minati De, Saksham Jain, Sarat Varma Kallepalli, and Satyam Singh. Online geometric covering and piercing. *Algorithmica*, 86(9):2739–2765, 2024. doi:10.1007/S00453-024-01244-1.

14 Minati De, Ratnadip Mandal, and Satyam Singh. New lower bound and algorithm for online geometric hitting set problem. In *COCOON*, volume 15983 of *Lecture Notes in Computer Science*, pages 234–247. Springer, 2025. doi:10.1007/978-981-95-0215-8_18.

15 Minati De, Satyam Singh, and Csaba D. Tóth. Online hitting sets for disks of bounded radii. In *ESA*, volume 351 of *LIPIcs*, pages 50:1–50:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ESA.2025.50.

16 Adrian Dumitrescu, Anirban Ghosh, and Csaba D. Tóth. Online unit covering in Euclidean space. *Theor. Comput. Sci.*, 809:218–230, 2020. doi:10.1016/j.tcs.2019.12.010.

17 Adrian Dumitrescu and Csaba D. Tóth. Online unit clustering and unit covering in higher dimensions. *Algorithmica*, 2022. doi:10.1007/s00453-021-00916-6.

18 Guy Even, Dror Rawitz, and Shimon Moni Shahar. Hitting sets when the VC-dimension is small. *Information Processing Letters*, 95(2):358–362, 2005. doi:10.1016/J.IPL.2005.03.010.

19 Guy Even and Shakhar Smorodinsky. Hitting sets online and unique-max coloring. *Discrete Applied Mathematics*, 178:71–82, 2014. doi:10.1016/J.DAM.2014.06.019.

20 Uriel Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998. doi:10.1145/285055.285059.

21 Arindam Khan, Aditya Lonkar, Saladi Rahul, Aditya Subramanian, and Andreas Wiese. Online and dynamic algorithms for geometric set cover and hitting set. In *SoCG*, pages 46–1, 2023. doi:10.4230/LIPIcs.SOCG.2023.46.

22 Nabil H Mustafa. Computing optimal epsilon-nets is as easy as finding an unhit set. In *ICALP*, pages 87–1, 2019. doi:10.4230/LIPIcs.ICALP.2019.87.

23 Nabil H Mustafa. *Sampling in combinatorial and geometric set systems*, volume 265. American Mathematical Society, 2022.

24 Nabil H Mustafa and Kasturi Varadarajan. Epsilon-approximations & epsilon-nets. In *Handbook of Discrete and Computational Geometry*, pages 1241–1267. Chapman and Hall/CRC, 2017.

25 János Pach and Pankaj K Agarwal. *Combinatorial geometry*. John Wiley & Sons, 2011.

26 Kasturi Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *STOC*, pages 641–648, 2010. doi:10.1145/1806689.1806777.

Euclidean Noncrossing Steiner Spanners of Nearly Optimal Sparsity

Sujoy Bhore
Department of Computer Science & Engineering, Indian Institute of Technology Bombay, Mumbai, India

Sándor Kisfaludi-Bak
Aalto University, Espoo, Finland

Lazar Milenković
Tel Aviv University, Israel
INSAIT, Sofia University "St. Kliment Ohridski", Bulgaria

Csaba D. Tóth
Department of Mathematics, California State University Northridge, Los Angeles, CA, USA
Department of Computer Science, Tufts University, Medford, MA, USA

Karol Węgrzycki
Max Planck Institute for Informatics, Saarbrücken, Germany

Sampson Wong
University of Copenhagen, Denmark

Abstract

A Euclidean noncrossing Steiner $(1+\varepsilon)$-spanner for a point set $P \subset \mathbb{R}^2$ is a planar straight-line graph that, for any two points $a, b \in P$, contains a path whose length is at most $1+\varepsilon$ times the Euclidean distance between a and b. We construct a Euclidean noncrossing Steiner $(1+\varepsilon)$-spanner with $O(n/\varepsilon^{3/2})$ edges for any set of n points in the plane. This result improves upon the previous best upper bound of $O(n/\varepsilon^4)$ obtained nearly three decades ago. We also establish an almost matching lower bound: There exist n points in the plane for which any Euclidean noncrossing Steiner $(1+\varepsilon)$-spanner has $\Omega_\mu(n/\varepsilon^{3/2-\mu})$ edges for any $\mu > 0$. Our lower bound uses recent generalizations of the Szemerédi-Trotter theorem to disk-tube incidences in geometric measure theory.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners; Theory of computation → Routing and network design problems; Mathematics of computing → Graph algorithms

Keywords and phrases geometric network design, spanners, crossing number, incidences

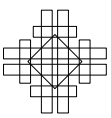
Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.15

Related Version *Full Version*: https://arxiv.org/abs/2602.17801v1

Funding *Sujoy Bhore*: Work supported in part by ANRF ARG-MATRICS, Grant 002465.
Sándor Kisfaludi-Bak: Supported by the Research Council of Finland, Grant 363444.
Lazar Milenković: Funded by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, and the United States National Science Foundation (NSF). This research was partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).
Csaba D. Tóth: Research supported, in part, by the NSF award DMS-2154347.
Karol Węgrzycki: Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) grant number 559177164.
Sampson Wong: Supported by the European Union's Marie Skłodowska-Curie Actions Postdoctoral Fellowship, grant number 101146276.



© Sujoy Bhore, Sándor Kisfaludi-Bak, Lazar Milenković, Csaba D. Tóth, Karol Węgrzycki, and Sampson Wong;
licensed under Creative Commons License CC-BY 4.0





1 Introduction

Spanners are a classical tool for data compression in graphs and network optimization. Formally, a t-spanner for an edge-weighted graph $G = (V, E, w)$ and a (stretch) parameter $t \geq 1$, is a subgraph H of G in which the shortest-path distance between any two vertices in V is at most t times larger than in G [3]. *Metric spanners* can approximate distances in a finite metric space (X, d) by setting $V = X$ and the edge weights to be the metric distances between the vertices. Geometric networks [69] are an important class of metric spanners, with applications in the design of physical networks in low-dimensional Euclidean spaces. Results in Euclidean spaces are also applicable for other settings via metric embeddings into Euclidean spaces [5, 28, 66, 67].

For Euclidean spanners in the plane, research efforts have diverged into two distinct regimes: (1) $(1+\varepsilon)$-*spanners*, where the stretch $t = 1+\varepsilon$ is arbitrarily close to 1, and the minimum size and weight of a $(1+\varepsilon)$-spanner is bounded by a function of $1/\varepsilon$ [63, 10, 61, 64], and (2) *plane* spanners, where the edges of the spanner are noncrossing line segments in $\mathbb{R}^2$ [27, 24, 56, 26, 46]. Researchers have made strides in both regimes over the last decade, and have uncovered optimal or near-optimal trade-offs between key parameters (see details below). However, very little attention was given to spanners that meet a dual objective: $1+\varepsilon$ stretch for arbitrarily small $\varepsilon > 0$ *and* noncrossing edges in $\mathbb{R}^2$.

This paper focuses on this dual objective. The problem may have avoided scrutiny because it has seemingly trivial answers: On the one hand, a simple instance of four points at the vertices of a square shows that a noncrossing straight-line Euclidean spanner cannot achieve stretch less than $\sqrt{2}$. If we insist on noncrossing edges and stretch close to 1, then we must allow *Steiner points*. On the other hand, if Steiner points are allowed, then the planarization of an optimal $(1+\varepsilon)$-spanner (by introducing Steiner vertices at edge crossings) provides a *noncrossing* Steiner $(1+\varepsilon)$-spanner with the same weight. However, planarization substantially increases the number of edges (hence the size of the spanner). Our goal is to find the best trade-offs between $\varepsilon > 0$ and the number of Steiner points for a Euclidean noncrossing Steiner $(1+\varepsilon)$-spanner in $\mathbb{R}^2$.

▶ **Problem 1.** *Determine $s(n, \varepsilon)$, defined as the minimum integer such that every set of n points in Euclidean plane admits a Euclidean noncrossing Steiner $(1+\varepsilon)$-spanner with at most $s(n, \varepsilon)$ Steiner points.*

This problem was also posed as Open Problem 17 in a survey by Bose and Smid [27, Section 4] in 2013. The best bound available at that time was given by Arikati et al. [9]: They constructed a Euclidean noncrossing Steiner $(1+\varepsilon)$-spanner with $O(n/\varepsilon^4)$ Steiner points by taking rectangular decompositions for n points in $O(1/\varepsilon)$ equally spaced directions.

Previous work. There are several possible approaches to address Problem 1. For n points in the plane, there are $(1+\varepsilon)$-spanners with $O(n/\varepsilon)$ edges (e.g., θ-graphs [37]); and this bound is the best possible. Since the edges of a spanner may pairwise cross, a naïve analysis of straightforward planarization would lead to a Euclidean Steiner $(1+\varepsilon)$-spanner with $O(n^2/\varepsilon^2)$ Steiner points.

Alternatively, one could try to bound the number of edge crossings in a $(1+\varepsilon)$-spanner (without Steiner points). Given a set $P \subset \mathbb{R}^2$ of n points, the greedy $(1+\varepsilon)$-spanner of Althöfer et al. [7] is constructed as follows: sort the $\binom{n}{2}$ possible edges by nondecreasing length, initialize an empty graph $H = (P, \emptyset)$, and add an edge ab to H if $d_H(a, b) > (1+\varepsilon) \cdot d(a, b)$. The greedy $(1+\varepsilon)$-spanner has $O(n/\varepsilon)$ edges [7, 30, 31]; and Eppstein and Khodabandeh [49]

proved that every edge ab crosses $O(1/\varepsilon^8)$ edges that are longer than ab. Consequently, it has $O(n/\varepsilon^9)$ crossings: planarization would create this many Steiner points. This bound is weaker than the previous bound of $O(n/\varepsilon^4)$ by Arikati et al. [9].

For n points in the plane and $k \in \mathbb{N}$, the *emanation graph of grade* k, introduced by Hamedmohseni et al. [54], is constructed by shooting 2^{k+1} rays from each given point, where the shorter rays stop the longer ones upon collision. The emanation graph of grade k is a noncrossing Steiner spanner with $O(2^k n)$ Steiner points. Hamedmohseni et al. [54] show that the stretch factor is at most $\sqrt{10}$, but for every $k \in \mathbb{N}$, there are point sets for which the stretch factor is arbitrarily close to $\sqrt{2}$; hence this approach does not lead to $(1+\varepsilon)$-spanners.

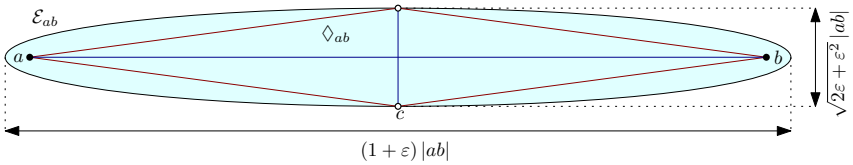


Figure 1 An ellipse $\mathcal{E}_{ab}$ with foci a and b, and major axis of length $(1+\varepsilon)\,|ab|$, and rhombus $\Diamond_{ab}$.

Cone-restricted spanners. Let P be a set of n points in the plane. For any $a, b \in P$, any ab-path of length at most $(1+\varepsilon)\,|ab|$ lies in an ellipse $\mathcal{E}_{a,b}$ with foci a and b and major axis $(1+\varepsilon)\cdot|ab|$; see Figure 1. Note that for small $\varepsilon > 0$, the ellipse $\mathcal{E}_{ab}$ is long and narrow. It is known that in every ab-path of length at most $(1+\varepsilon)\,|ab|$, the total length of the edges that make an angle $\alpha \le O(\sqrt{\varepsilon})$ with the line segment ab is $\Omega(|ab|)$ [18]. The angle threshold $\alpha \le O(\sqrt{\varepsilon})$ is the best possible: For example, if c is an intersection point of $\mathcal{E}_{ab}$ and its minor axis, the ab-path (a, c, b) has length $|ac| + |cb| = (1+\varepsilon)\,|ab|$, but both ac and cb make an angle of $\Theta(\sqrt{\varepsilon})$ with ab. We define a variant of Euclidean $(1+\varepsilon)$-spanners (with or without Steiner points), where we require an ab-path, for all $a, b \in P$, in which *all* edges make an angle $O(\sqrt{\varepsilon})$ with the line segment ab.

▶ **Definition 2.** *Let P be a set of n points in $\mathbb{R}^d$, for constant dimension $d \in \mathbb{N}$, and let $\alpha \in (0, \pi/2]$. A (Steiner) graph $G = (V, E)$, with $P \subseteq V$, is a* cone-restricted *(Steiner) $(1+\varepsilon)$-spanner for P if for every $a, b \in P$, there is an ab-path $(a = p_0, p_1, \ldots, p_m = b)$ in G such that $\sphericalangle(\overrightarrow{ab}, \overrightarrow{p_{i-1}p_i}) \le \sqrt{\varepsilon}$ for all $i = 1, \ldots, m$.*

Note that if G is a cone-restricted (Steiner) $(1+\varepsilon)$-spanner for P, then for every $a, b \in P$, there is an ab-path of length at most $(1+\varepsilon)\,|ab|$ that lies in the rhombus $\Diamond_{ab}$ spanned by a, b, and the two intersection points of $\mathcal{E}_{ab}$ with its minor axis; see Lemma 9.

1.1 Contributions and technical highlights

Upper bound. Our first contribution is a noncrossing Steiner $(1+\varepsilon)$-spanner with $O(n/\varepsilon^{3/2})$ Steiner vertices. This improves upon the previous best result by Arikati et al. [9], which has $O(n/\varepsilon^4)$ Steiner vertices.

▶ **Theorem 3.** *For every $\varepsilon > 0$ and every set of n points in Euclidean plane, there is a noncrossing Steiner $(1+\varepsilon)$-spanner with $O(n/\varepsilon^{3/2})$ Steiner vertices. Furthermore, there is such a spanner that is cone-restricted, and can be computed in $O((n \log n)/\varepsilon^{3/2})$ time.*

Arikati et al. [9] construct a set of noncrossing graphs G_i for $1 \le i \le k$. The final spanner is $\bigcup_{i=1}^{k} G_i$, where a Steiner vertex is added at every edge crossing between G_i and G_j, $i \ne j$. Our construction improves on the construction of Arikati et al. [9] in several ways. First, we

use fewer graphs G_i when constructing our spanner. Specifically, we use $k = O(1/\sqrt{\varepsilon})$ instead of $k = O(1/\varepsilon)$ graphs. Whereas the previous construction [9] uses $O(1/\varepsilon)$ rotated copies of the point set to approximate the L_2 distance with the L_1 distance in one of the copies, we instead use $O(1/\sqrt{\varepsilon})$ carefully chosen linear transformations to achieve cone-restricted paths between all pairs of points. Second, our graphs G_i are of smaller size than those in Arikati et al. [9]. Both constructions obtain G_i by refining the Balanced Box Decomposition into an axis-parallel spanner under the L_1 metric. We are able to use the properties of our linear transformations to obtain the same stretch guarantee using a coarser refinement of the Balanced Box Decomposition. In particular, we obtain $|G_i| = O(n)$, improving on $|G_i| = O(n/\varepsilon^2)$ in [9]. Third, our construction fills some details that are missing from both [9] and [27, Chapter 4].

Lower bounds. Our second contribution is an almost matching lower bound on the number of Steiner vertices in any Euclidean noncrossing Steiner $(1+\varepsilon)$-spanner.

▶ **Theorem 4.** *For every sufficiently small $\varepsilon, \mu > 0$ and every $n \in \mathbb{N}$, there exists a set of n points in the Euclidean plane for which every noncrossing Steiner $(1+\varepsilon)$-spanner has $\Omega_\mu(n/\varepsilon^{3/2-\mu})$ Steiner vertices, where the constant hidden in the $\Omega_\mu(.)$ notation depends only on μ.*

To the best of our knowledge, the best previous unconditional lower bound follows from the known size lower bound of $\Omega(n/\sqrt{\varepsilon})$ for Euclidean Steiner $(1+\varepsilon)$-spanners [63], which also applies to noncrossing spanners. In this paper, the canonical example to prove lower bounds is the following construction: consider the unit square $[0,1]^2$ and let A (resp., B) be the set of equally spaced points on the left (resp., right) side of $[0,1]^2$ so that the distance between any two consecutive points in A (resp., B) is $4\sqrt{\varepsilon}$. The point set P is the union of points in A and B; see Figure 2 for an illustration. Observe that: (i) the minimum pairwise distance in P is $4\sqrt{\varepsilon}$, (ii) the maximum pairwise distance (a.k.a., diameter) in P is $\sqrt{2}$, and (iii) for any $a \in A$ and $b \in B$, the slope of the segment ab is between -1 and 1. Let G be a noncrossing Steiner $(1+\varepsilon)$-spanner with the minimal number of Steiner points for P.

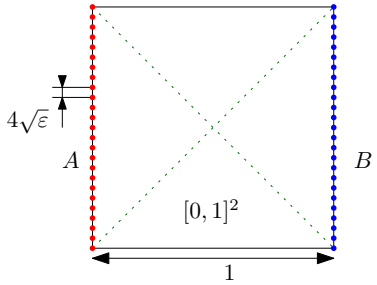


Figure 2 Point set $A \cup B$, where A and B lie on two opposite sides of a unit square.

First, we sketch key ideas behind a weaker $\Omega(n/\varepsilon)$ lower bound (the proof can be found in the full version of the paper).

▶ **Theorem 5.** *For every sufficiently small $\varepsilon > 0$ and every $n \in \mathbb{N}$, there exists a set of n points in the Euclidean plane for which every noncrossing Steiner $(1+\varepsilon)$-spanner has $\Omega(n/\varepsilon)$ Steiner vertices.*

Let $A_0 \subseteq A$ consist of every third point from the bottom third of the left side, and $B_0 \subseteq B$ consist of every third point from the top third of the right side. This ensures $|A_0|, |B_0| = \Omega(1/\sqrt{\varepsilon})$ and for each pair $(a,b) \in A_0 \times B_0$ the segment ab has slope in $[\frac{1}{3}, 1]$.

For each pair $(a, b) \in A_0 \times B_0$, let γ_{ab} denote a shortest path in G from a to b. We apply the result of Bhore and Tóth [18, Lemma 4] to conclude that the set E_{ab} of edges in γ_{ab} having angle at most $3\sqrt{\varepsilon}$ with ab satisfies $\|E_{ab}\| \geq \frac{7}{9}|ab|$, i.e., the total length of the edges in E_{ab} is at least $\frac{7}{9}|ab|$. Moreover, using a geometric argument, we can show that for distinct pairs $(a, b), (a', b') \in A_0 \times B_0$, the sets E_{ab} and $E_{a'b'}$ are pairwise disjoint.

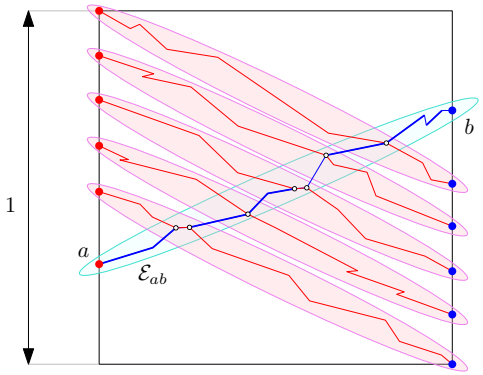


Figure 3 An ab-path and its intersections with other paths between point pairs of slope $-1/2$.

Now, consider the collection of all ellipses $\mathcal{E}_{cd}$ with $c, d \in A \cup B$ and slope $-\frac{1}{2}$ (see Figure 3). As these ellipses are pairwise disjoint (Lemma 10), each such ellipse creates a Steiner vertex wherever γ_{ab} (for $(a, b) \in A_0 \times B_0$) intersects it. Since there are $\Theta(1/\sqrt{\varepsilon})$ such disjoint ellipses, and each path γ_{ab} must cross $\Theta(1/\sqrt{\varepsilon})$ of them, each path γ_{ab} contains $\Theta(1/\sqrt{\varepsilon})$ Steiner points in these intersections. The bound on the total length $|\gamma_{ab}| \leq (1+\varepsilon)|ab| < 2$ combined with the pigeonhole principle implies that the average length of an edge in E_{ab} is $O(\sqrt{\varepsilon})$. Consequently, E_{ab} contains at least $\Omega(1/\sqrt{\varepsilon})$ edges. Since there are $\Omega(1/\varepsilon)$ pairs in $A_0 \times B_0$ and the sets E_{ab} are edge-disjoint, the total number of edges is $\Omega(1/\varepsilon) \cdot \Omega(1/\sqrt{\varepsilon}) = \Omega(1/\varepsilon^{3/2}) = \Omega(n/\varepsilon)$ for the basic construction. See the full version for a detailed proof.

This lower bound is already stronger than the lower bound of $\Omega(n/\sqrt{\varepsilon})$ that can be derived from the crossing case [63]. However, the limitation of this approach is the fact that the pigeonhole argument can give only $\Omega(1/\sqrt{\varepsilon})$ edges per path γ_{ab}. To improve this to $\Omega_\mu(n/\varepsilon^{3/2-\mu})$ for any $\mu > 0$, we will use techniques from geometric measure theory (see the full version for the full proof of Theorem 4). To use these tools, we first need to show that most of the spanner paths have certain properties. Let M be the square of side-length $1/8$ in the middle of $[0, 1]^2$. We decompose M into an $O(1/\sqrt{\varepsilon}) \times O(1/\sqrt{\varepsilon})$ grid consisting of square *windows* and analyze the structure of spanner paths within each window. We will also restrict our attention to ellipses $\mathcal{E}_{ab}$ where the corresponding segment ab has slope $\lambda \in [1/4, 3/4]$, and we say that such ellipses and spanner paths are in the *positive bundle*, or ab has slope $\lambda \in [-3/4, -1/4]$, where the corresponding ellipses and spanner paths belong to the *negative bundle*. The key properties we establish are:

- Each window W contains crossing ellipses for each slope in the positive and negative bundles.
- Each spanner path γ_{ab} is *adventurous* in at most $c/\sqrt{\varepsilon}$ windows, meaning it goes outside a narrow strip $R_W(\gamma_{ab})$ of width $O(\varepsilon)$ in only a small number of windows. Here c is a small constant.
- Each spanner path γ_{ab} is *skewed* in at most $c/\sqrt{\varepsilon}$ windows, meaning its direction deviates significantly from the direction of the segment ab in only a small number of windows. Again, c is a small constant.

- More than half of the windows in M are *well-behaved*, meaning they are crossed by sufficiently many non-adventurous, non-skewed spanner paths from both the positive and negative bundles.

In a well-behaved window W, we can also identify collections Ψ_W^+ and Ψ_W^- of spanner paths such that each path $\gamma \in \Psi_W^+ \cup \Psi_W^-$ is non-adventurous, non-skewed, and the paths and their corresponding strips $R_W(\gamma)$ of width $O(\varepsilon)$ have the following properties (see Figure 4):

(i) Paths in Ψ_W^+ and Ψ_W^- pairwise cross within the window.

(ii) The directions of the strips $R_W(\gamma)$ corresponding to these paths differ by at least $\Omega(\sqrt{\varepsilon})$.

(iii) The strips corresponding to distinct paths have small intersection areas, at most half the area of any individual strip.

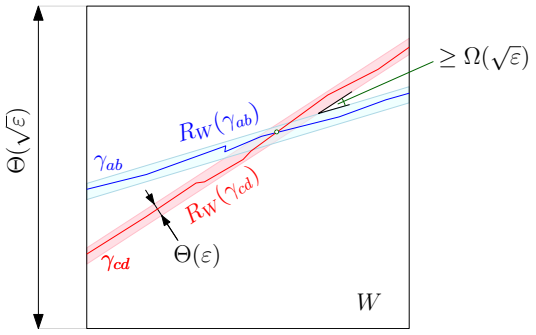


Figure 4 A well-behaved window W with two non-adventurous non-skewed paths $\gamma_{ab}, \gamma_{cd} \in \Psi_W^+$.

After this preparation, we want to analyze the number of Steiner vertices. A natural tool would be the classical Szemerédi–Trotter theorem which states that for any set of ℓ lines in the plane, the number of points incident to at least r of these lines is at most $O(\ell^2/r^3 + \ell/r)$. Ideally, we would like to apply this theorem to count the number of Steiner vertices incident to many spanner paths. However, the Szemerédi–Trotter theorem does not directly apply, since our spanner paths are not straight lines.

Motivated by problems in geometric measure theory, the Szemerédi–Trotter theorem was generalized to *disk-tube incidences* [53]. Instead of points and lines, we count intersections between δ-disks (small disks of radius δ) and δ-tubes (long rectangles of width δ). Fu, Gan and Ren [52] recently proved that only $O_\mu(|\mathcal{T}|^2/r^3)$ disjoint r-rich disks (each intersecting at least r tubes) exist when tubes in $\mathcal{T}$ are sufficiently well-spaced and separated in direction.[1]

We apply their theorem to the tubes formed by the strips $R_W(\gamma_{ab})$ corresponding to the spanner paths with $\delta = \Theta(\sqrt{\varepsilon})$ for every well-behaved window W (see Figure 4). Properties (i), (ii) and (iii) guarantee that the collection of tubes in each well-behaved window W satisfies the spacing and direction separation conditions required by the result of Fu, Gan and Ren [52].

The theorem of Fu, Gan and Ren [52] implies that only $O_\mu(1/(\varepsilon r^3))$ disjoint r-rich δ-disks can intersect tubes in our collection. In their result, there is a technical requirement that $r > \delta^{1-2\mu}|\mathcal{T}_{\max}|$, and so the smallest value of r we can choose is $r = \Omega_\mu(1/\varepsilon^\mu)$. This means that most of the $\Theta(1/\varepsilon)$ crossing points are not covered by the r-rich disks. Therefore, there must exist $\Omega_\mu(1/\varepsilon)$ crossing points where the Steiner vertex is incident to fewer than

[1] The exact definition of *sufficiently well-spaced* is quite technical and is guided by the $\mu > 0$ parameter (see the full version for details).

$r_0 = \Theta_\mu(1/\varepsilon^\mu)$ spanner paths from our tube collection. Counting the tube crossings at these low-degree Steiner vertices yields at least $\Omega_\mu(1/\varepsilon)/r_0^2 = \Omega_\mu(1/\varepsilon^{1-2\mu})$ such vertices per window, and summing over all $\Omega(1/\varepsilon)$ well-behaved windows gives the $\Omega_\mu(1/\varepsilon^{2-2\mu})$ lower bound on the number of Steiner vertices in G. Since $n = \Theta(1/\sqrt{\varepsilon})$, this implies the desired lower bound of $\Omega_\mu(n/\varepsilon^{3/2-2\mu})$. Finally, we scale μ by a factor of 2. This concludes the sketch of the proof of Theorem 4, see the full version for details.

Theorem 4 leaves a small gap of roughly $\varepsilon^{o(1)}$ between our upper and lower bounds. We are able to show that this gap can be closed if we restrict our attention to cone-restricted spanners which were used in the construction of Theorem 3. The proof of Theorem 6 uses a generalization of the celebrated Crossing Lemma to so-called degenerate crossings [2, 70].

▶ **Theorem 6.** *For every sufficiently small $\varepsilon > 0$ and every $n \in \mathbb{N}$, there exists a set of n points in the plane for which every cone-restricted plane Steiner $(1+\varepsilon)$-spanner has $\Omega(n/\varepsilon^{3/2})$ Steiner vertices. Up to constant factors, this lower bound is the best possible (cf. Theorem 3).*

Organization. After covering further related works (Section 1.2) and the preliminaries (Section 2), we provide the construction for our sparse noncrossing Steiner spanner (Section 3). Due to space limitations, complete proofs are deferred to the full version. The proofs of the lower bounds (Theorems 4–6) are also deferred to the full version.

1.2 Further related previous work

As noted above, Euclidean spanners have been studied under two independent regimes:

$(1+\varepsilon)$-Spanners. The first Euclidean $(1+\varepsilon)$-spanners for arbitrary $\varepsilon > 0$ were obtained independently by Clarkson [37] and Keil [57], and these works also introduced the fixed-angle Θ-graph (a close variant of the Yao graph [81]) as a basic construction tool. Ruppert and Seidel [76] extended this to $\mathbb{R}^d$, by giving $(1+\varepsilon)$-spanners with $O(n/\varepsilon^{d-1})$ edges for constant d. Le and Solomon [63] proved that this dependence on ε is tight: for every $\varepsilon > 0$ and constant d, there exist point sets in $\mathbb{R}^d$ for which any $(1+\varepsilon)$-spanner must have $\Omega(\varepsilon^{-(d-1)})$ edges whenever $\varepsilon = \Omega(n^{-1/(d-1)})$.

Another key parameter of a spanner is its *lightness*. For a set of points, it is the ratio between its total edge weight and the weight of the MST of the point set. Building on the greedy spanner of Althöfer et al. [7], Das et al. [41] proved that this construction achieves constant lightness and stretch $1+\varepsilon$ in $\mathbb{R}^3$, which was later extended to all $\mathbb{R}^d$ by Das et al. [42]. Rao and Smith [72] established that the greedy $(1+\varepsilon)$-spanner in $\mathbb{R}^d$ has lightness $1/\varepsilon^{O(d)}$, and a long line of refinements culminated in the bound $O(\varepsilon^{-d}\log(1/\varepsilon))$ by Le and Solomon [63]. Besides achieving small *stretch* and *sparsity*[2], spanners are often required to satisfy additional desirable properties such as bounded degree or diameter. There has been extensive work on understanding the optimal trade-offs among these parameters for Euclidean spanners (see, e.g., [10, 17, 47, 45, 61]).

Euclidean Steiner spanners. Steiner points can significantly reduce the weight of distance approximation structures (see, e.g., [48, 77]). Note that a Steiner t-spanner for a point set P must guarantee stretch t only for point pairs in P. Le and Solomon [62] constructed Steiner $(1+\varepsilon)$-spanners of sparsity $O(\varepsilon^{(1-d)/2})$ in $\mathbb{R}^d$ for all $d \geq 2$, and this bound is the

[2] The sparsity of a spanner is the ratio of the number of edges in the spanner to the size of MST.

best possible [18]. In the plane, Bhore and Tóth [18] constructed Steiner $(1+\varepsilon)$-spanners of lightness $O(1/\varepsilon)$, this bound is also tight [63]. In dimensions $d \geq 3$, the current best upper and lower bounds for lightness are $\widetilde{O}(1/\varepsilon^{(d+1)/2})$ and $\Omega(1/\varepsilon^{d/2})$ [18, 63]. Recent work has also analyzed online algorithms for Euclidean Steiner spanners and obtained several asymptotically tight bounds [15, 19].

Noncrossing spanners. Chew [33] first proved the existence of a Euclidean spanner with stretch $\sqrt{10}$ and $O(n)$ noncrossing edges, the stretch was later improved to 2 [34]. Keil and Gutwin [58] showed that the Delaunay triangulation is a 2.42-spanner. Later, Bonichon et al. [23] gave tight bounds of $\sqrt{4+2\sqrt{2}}$ for the L_1- and L_∞-Delaunay graphs. Subsequently, Bose et al. [25] showed that the Yao graph is a $8\sqrt{2}$-spanner. See the comprehensive survey by Bose and Smid [27] for results on plane spanners up to 2013. Since then, several core questions raised in that survey on plane spanners have seen notable progress. The long-open existence of bounded-degree plane spanners has been resolved for degree 4 through constructions of Bonichon et al. [24]. Later, Kanj et al. [56] obtained improved stretch bounds. Bose et al. [26] further tightened the trade-off by giving an algorithm to construct degree-8 plane spanners with stretch ≈ 4.414. Dumitrescu and Ghosh [46] strengthened lower bounds on plane-spanner dilation. For degree 3, progress has been obtained for restricted families of point sets, such as points in convex position [20]. The framework has also been extended to constrained visibility, where van Renssen and Wong [79] showed that visibility graphs among polygonal obstacles admit noncrossing spanners with bounded degree and constant stretch (see also [16, 1] for some recent work on plane spanners in polygonal domains). Further developments include refined stretch analysis for planar variants of θ-like geometric graphs [26], as well as general geometric conditions under which sweepline-based constructions produce planar spanners [65].

Previous work related to cone-restricted spanners. We have defined *cone-restricted* $(1+\varepsilon)$-spanners for finite point sets in the plane (Definition 2). Similar concepts have previously been used for other purposes. In a *geometric graph* $G = (V, E)$, the vertices are distinct points in the plane and the edges are straight-line segments. A geometric graph is *strongly monotone* [8, 59, 51] if for every $a, b \in V$, there is a ab-path $(a = v_0, v_1, \ldots, v_m = b)$ in which $\sphericalangle(\overrightarrow{ab}, \overrightarrow{v_{i-1}v_i}) \leq \pi/2$ for all $i = 1, \ldots, m$. However, this property does not guarantee any stretch factor.

A geometric graph is *angle-monotone with width* γ [21, 43, 68] if for every $a, b \in V$, there is an ab-path in which the angle between any two edges is at most γ. For example, the axis-aligned grid graph induced by n points in the plane is angle monotone with width $(\pi/2)$, and is a $\sqrt{2}$-spanner, however, it uses $O(n^2)$ Steiner vertices. Dehkordi et al. [43] constructed, for n points in $\mathbb{R}^2$, a plane angle-monotone graph of width $\pi/2$ using $O(n)$ Steiner points. Bonichon et al. [21] showed that the half-θ_6 graph [22] is angle-monotone with width $2\pi/3$, and $O(n)$ edges, however, it is not necessarily planar. Lubiw and Mondal [68] constructed an angle monotone graph with width $\pi/2$ with $O(n^2 \log\log n/\log n)$ edges without Steiner vertices (but with crossings). They also consider a version of the problem with Steiner vertices, however, they require the angle-monotone property for all pairs of vertices (including Steiner vertices), and allow additional crossings. For every $\gamma > 0$ and every set P of n points in the plane, they construct a Steiner angle-monotone graph of width γ with $O(\frac{n}{\gamma} \log \Phi(P))$ edges, where $\Phi(P)$ is an unbounded parameter that depends on the point configurations.

Other constraints imposed on ab-path include *greedy* [66, 71], *self-approaching* [6] and *increasing-chord* [40] properties: A geometric graph $G = (V, E)$ is *greedy* if for every $a, b \in V$, there is an ab-path $(a = v_0, v_1, \ldots, v_m = b)$ that monotonically gets closest to b, that is,

$d(v_i, b) < d(v_{i-1}, b)$ for all $i = 1, \ldots, m$. It is *self-approaching* if $d(v_j, v_k) \leq d(v_i, v_k)$ for all $0 \leq i < j < k \leq m$; and *increasing-chord* if there is an ab-path that is self-approaching in both directions. The self-approaching and increasing-chord properties imply a stretch factor of 5.34 [55], at most $2\pi/3$ [74] (see also [4]), resp., while the greedy property alone does not imply any stretch guarantee.

We also mention a couple of concepts that sound similar to cone-restricted spanners, but are different. The classical θ- and Yao-graphs are constructed by connecting every vertex v to a "closest" point in cones of apex v and aperture θ. They are known to be $O(\theta)$-spanners for $\theta \leq \pi/2$, but the ab-paths of length $O(\theta) \cdot d(a, b)$ are not necessarily cone-restricted: They may contain (short) edges that make an arbitrary angle with the line segment ab.

Another concept, under a similar name, was introduced by Carmi and Smid [29]: A geometric graph is *θ-angle-constrained* if for every vertex $v \in V$ the angle between any two edges incident to v is *at least* θ. They note that the classical greedy $(1+\varepsilon)$-spanner by Althöfer et al. [7] is $\Omega(\varepsilon)$-angle constrained. For every $\theta \in (0, \pi/3)$, and n points in the plane, one can construct a θ-angle-constrained $(1 + O(\theta))$-spanner in $O(n \log n)$ time [29]; and this is not always possible for $\theta > \pi/3$ [12].

Connections to incidences and geometric measure theory. Szemerédi and Trotter [78] proved that n points and ℓ lines in $\mathbb{R}^2$ determine $O(n^{2/3}\ell^{2/3} + n + \ell)$ point-line incidences, and this bound is the best possible [50]. Motivated by connections to geometric measure theory [80] significant progress was made on a generalization to disk-tube incidences, which is the number of intersections between well-spaced disks of radius δ and *δ-tubes*, where a δ-tube is the δ-neighborhood of a line, in a unit square $[0, 1]^2$; see [39, 44, 52, 53].

The number of disk-tube incidences (used in multiple scales) was instrumental in several recent breakthroughs in combinatorial geometry and geometric measure theory. For example, Heilbronn's classical problem [75] asks for the minimum $h(n) > 0$ such that any set P of n points in the unit square $[0, 1]^2$ determines a nondegenerate triangle $\Delta(abc)$ of area at most $h(n)$. A line segment ab is the base of a triangle $\Delta(abc)$ of area A if and only if there exists a point $c \in P$ in the $\frac{2A}{d(a,b)}$-neighborhood of the line spanned by ab. Cohen et al. [38, 39, 82] recently proved $h(n) \geq \Omega(n^{-8/7-1/2000})$, improving on the previous bound $h(n) \geqq \Omega(n^{8/7})$ by Komlós et al. [60]. In another recent breakthrough using this machinery, Ren and Wang [73] completely solved the Furstenberg set problem.

2 Preliminaries

Balanced Box Decomposition. In our upper bound construction, we will use the Balanced Box Decomposition (BBD) of Arya et al. [11]. Given a set of points P, the BBD partitions the bounding box of P into a set of tiles, such that each tile is either a rectangle or defined by an outer rectangle and a sticky inner rectangle.

▶ **Definition 7** (Sticky). *In $\mathbb{R}^1$, an inner interval of width w is sticky with respect to an outer interval if its distances to the endpoints of the outer interval are either $= 0$ or $\geq w$. In $\mathbb{R}^d$, an inner box is sticky with respect to an outer box if their projections onto each of the d coordinate axes are sticky.*

▶ **Theorem 8** (BBD [11]). *Given a set of points P, one can partition the bounding box of P into $O(n)$ tiles such that*

(a) *each tile is either a rectangle, or an outer rectangle with a sticky inner rectangular hole,*
(b) *the rectangle, outer rectangle, and inner rectangle must have an aspect ratio of ≤ 3,*
(c) *each tile contains at most one point of P, moreover, this point lies on the tile's boundary.*

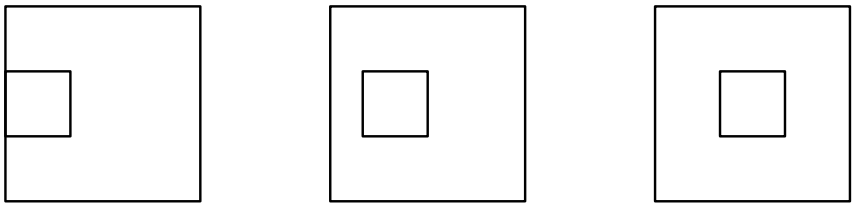


Figure 5 The middle figure shows an inner box that is not sticky, due to its x-projection.

Properties of cone-restricted spanners. We prove here that every cone-restricted (Steiner) $(1+\varepsilon)$-spanner is, in fact, a (Steiner) $(1+\varepsilon)$-spanner; which justifies calling them $(1+\varepsilon)$-spanners in Definition 2. (See also [21].)

▶ **Lemma 9.** *Let $a, b \in \mathbb{R}$, and let $\gamma = (a = p_0, p_1, \ldots, p_m = b)$ be a polygonal path such that $\sphericalangle(\overrightarrow{ab}, \overrightarrow{p_{i-1}p_i}) \leq \sqrt{\varepsilon}$ for all $i = 1, \ldots, m$. Then for $0 < \varepsilon \leq 1$,*

1. *the length of γ is bounded by $|\gamma| \leq (1+\varepsilon)\,|ab|$, and*
2. *$\gamma \subset \Diamond_{ab}$, where $\Diamond_{ab}$ is the rhombus $\Diamond_{ab}$ spanned by a, b, and the two intersection points of $\mathcal{E}_{ab}$ with its minor axis (cf. Figure 1).*

Basic point set for lower bounds. Our lower bounds are based on the same basic point set, which is known to give asymptotically tight bounds for lightness and sparsity for both Steiner and non-Steiner $(1+\varepsilon)$-spanners in the plane [18, 62, 63].

Let $\varepsilon \in (0, \frac{1}{16})$, and assume w.l.o.g. that $\varepsilon = 4^{-k}$ for some $k \in \mathbb{N}$. We first construct a point set P of size $|P| = 2(2^{k-2}+1) = \Theta(\varepsilon^{-1/2})$. Consider the unit square $[0,1]^2$. Let A be a set of $2^{k-2}+1$ equally spaced points on the left side of $[0,1]^2$; and B a set of $2^{k-2}+1$ equally spaced points on the right side of U. Our point set is $P = A \cup B$; see Figure 2.

Note that the minimum distance between any two points in A (resp., B) is $2^{2-k} = 4\sqrt{\varepsilon}$; the diameter of P is $\sqrt{2}$. Note also that for any $a \in A$ and $b \in B$, the segment ab makes an angle at most $\pi/4$ with a vertical line, in particular the absolute value of the slope of ab is at most 1.

We observe two easy properties of the point set $P = A \cup B$.

▶ **Lemma 10.** *If ab and $a'b'$ are parallel, then the ellipses $\mathcal{E}_{ab}$ and $\mathcal{E}_{a'b'}$ are disjoint.*

Next, we have a lower bound on the angle between two nonparallel segments ab and $a'b'$.

▶ **Lemma 11.** *For any $a, a' \in A$ and $b, b' \in B$, the following hold:*

(1) *if ab and $a'b'$ are parallel, then $\mathcal{E}_{ab}$ and $\mathcal{E}_{a'b'}$ are disjoint;*

(2) *if ab and $a'b'$ are nonparallel, then $\sphericalangle(ab, a'b') > 2\sqrt{\varepsilon}$.*

3 A sparse noncrossing Steiner spanner

Construction. Our construction is based on the noncrossing Steiner $(1+\varepsilon)$-spanner of Arikati et al. [9]. See also Section 4 in the survey by Bose and Smid [27].

We are given a set P of n points in the plane, and a parameter $\varepsilon > 0$. Let $k \in \mathbb{N}$. Note that $k = k(\varepsilon)$ and will be chosen later based on ε. We will construct a set of planar straight-line graphs G_i for $i \in \{1, \ldots, k\}$. Then we will construct the final spanner $G = \bigcup_{i=1}^{k} G_i$ as the union of the graphs G_i, where a Steiner point is inserted at each edge crossing between edges in G_i and G_j for $i \neq j$. For $i \in \{1, \ldots, k\}$, the graph G_i will be such that the edges have two

possible directions: they either make an angle of $i \cdot \frac{\pi}{k}$ or $(i+\delta) \cdot \frac{\pi}{k}$ with the positive x-axis, where δ is a positive integer[3]. Consider the affine transformation $T_i : \mathbb{R}^2 \to \mathbb{R}^2$ that maps unit vectors of direction $i \cdot \frac{\pi}{k}$ and $(i+\delta) \cdot \frac{\pi}{k}$ to unit vectors along the positive x- and y-axes, respectively. By applying the transformation on G_i, we obtain an axis-parallel graph $T_i(G_i)$ on the point set $T_i(P)$.

It remains to construct the axis-parallel graph $T_i(G_i)$. We use the Balanced Box Decomposition (BBD), which we introduced in Section 2. Recall that the BBD divides the bounding box of $T_i(P)$ into $O(n)$ tiles. For each of these $O(n)$ axis-aligned tiles, we will further subdivide the tile into at most $9K^2$ axis-aligned rectangles. If a tile contains no hole, we subdivide it into rectangles using K equally spaced horizontal lines and K equally spaced vertical lines. If a tile contains a hole, we first subdivide it into at most 9 rectangles using the four lines spanned by the four sides of the inner rectangle, and then subdivide each of these rectangles using K equally spaced horizontal lines and K equally spaced vertical lines. See Figure 6.

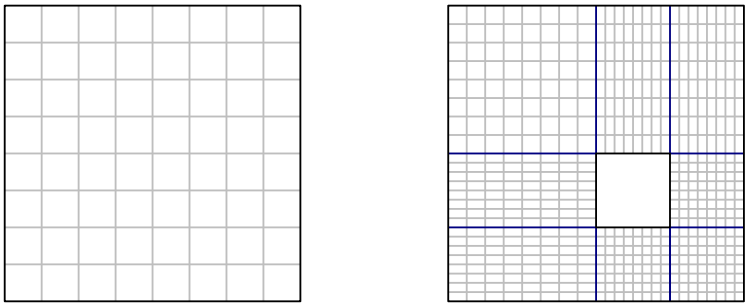


Figure 6 Further subdividing a tile into $\leq 9K^2$ rectangles, first by the four lines spanned by the sides of the inner rectangle (dark blue), and second by equally spaced axis-parallel lines (grey).

This yields a partition of the bounding box of $T_i(P)$ into $O(nK^2)$ axis-aligned rectangles such that each point in $T_i(P)$ lies on the boundary of one of the rectangles. This partition defines the axis-parallel graph $T_i(G_i)$. In particular, the vertices of the graph are the points $T_i(P)$ and the vertices of the rectangles in the partition, and the edges of the graph are the edges of the rectangles in the partition, or a pair of edges connected to a vertex of $T_i(G_i)$ if the vertex lies on an edge of a rectangle. This completes the construction of $T_i(G_i)$. We can apply the inverse transformation T_i^{-1} to obtain the graph G_i. Finally, by constructing the union $G = \bigcup_{i=1}^{k} G_i$ and adding Steiner points at edge crossings, we obtain the final graph.

A key difference between our construction and that of Arikati et al. [9] is that we use linear transformations, instead of rotated copies, of the BBD construction. This difference ultimately leads to our improved bound on the number of Steiner points. In particular, the properties of our linear transformations (Lemmas 12 and 13) allow us to use significantly fewer graphs and a significantly coarser refinement of the BBD, which correspond to smaller values for the parameters k and K, respectively.

Next, we define the parameters δ, k and K and compare them to [9]. Our linear transformations use $\delta = 3$, whereas the rigid motions of Arikati et al. [9] correspond to $\delta = \frac{k}{2}$. We use $k = O(\sqrt{1/\varepsilon})$ and $K = 100$, instead of the $k = O(1/\varepsilon)$, and $K = O(1/\varepsilon)$ used in [9]. Notably, we have $|G_i| = O(n)$ instead of $|G_i| = O(n/\varepsilon^2)$ in [9]. We will show that, even with fewer and smaller graphs G_i, we still obtain a $(1+\varepsilon)$-spanner.

[3] We choose $\delta = 3$, whereas Arikati et al. [9] choose $\delta = \frac{k}{2}$.

Stretch analysis. We will prove that G is a $(1+\varepsilon)$-spanner for the new construction, i.e., for the new values of k, δ and K. Let $a, b \in P$. Define $\sphericalangle ab$ to be the angle between the vector $\overrightarrow{ab}$ and the positive x-axis. For the remainder of this section, we will assume without loss of generality that $\sphericalangle ab \in [(i+1)\cdot\frac{\pi}{k}, (i+2)\cdot\frac{\pi}{k})$ where $i \in \{1, \ldots, k\}$. The next lemma is to prove that the transformation T_i sends vector $\overrightarrow{ab}$ into the angle class $[\frac{\pi}{12}, \frac{5\pi}{12})$. See Figure 7. The lemma assumes that ε is sufficiently small, and thus k is sufficiently large.

▶ **Lemma 12.** *If* $\sphericalangle ab \in [(i+1)\cdot\frac{\pi}{k}, (i+2)\cdot\frac{\pi}{k})$, *then* $\sphericalangle(T_i(a)T_i(b)) \in [\frac{\pi}{12}, \frac{5\pi}{12})$.

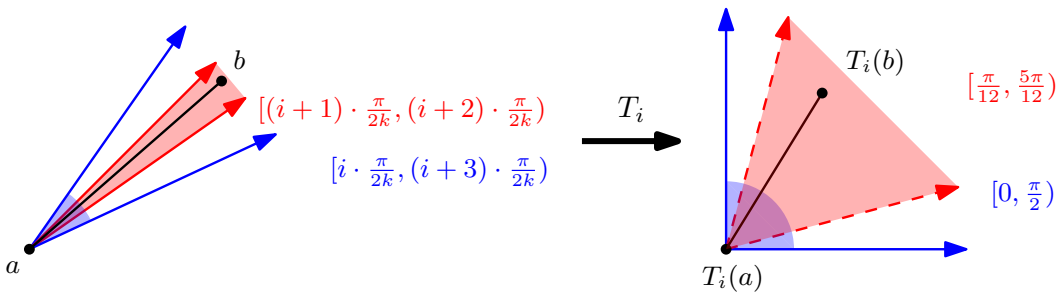


Figure 7 A visualization of the transformation T_i, which sends the unit vectors in the directions $i\cdot\frac{\pi}{k}$, $(i+\delta)\cdot\frac{\pi}{k}$ (left, blue) to the unit x- and y-vectors (right, blue). Lemma 12 states that the angle class $[(i+1)\cdot\frac{\pi}{k}, (i+2)\cdot\frac{\pi}{2k})$ (left, red) will be sent into the angle class $[\frac{\pi}{12}, \frac{5\pi}{12})$ (right, red).

Next, we use Lemma 12 to show that there is a staircase path in the graph $T_i(G_i)$.

▶ **Lemma 13.** *If* $\sphericalangle(T_i(a)T_i(b)) \in [\frac{\pi}{12}, \frac{5\pi}{12})$, *then there is an* xy*-monotone axis-parallel path from* $T_i(a)$ *to* $T_i(b)$ *in the graph* $T_i(G_i)$.

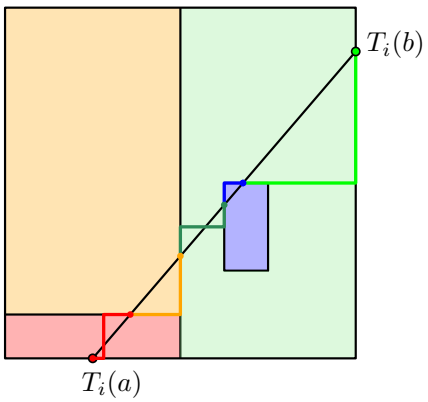


Figure 8 Replacing $T_i(a)T_i(b) \cap \tau$ with an xy-monotone axis-parallel path, for each tile τ.

Proof sketch. We trace the segment $T_i(a)T_i(b)$ across the tiles in the Balanced Box Decomposition of $T_i(P)$, and within each tile, we replace the segments in $T_i(a)T_i(b) \cap \tau$ with xy-monotone axis-parallel paths in $T_i(G_i)$. Let $cd \in T_i(a)T_i(b) \cap \tau$ be a segment connecting boundary points of τ. We have four cases, where the colors refer to Figure 8:

1. (Orange, Blue) c and d are on the outer boundary and cd connects adjacent sides of τ.
2. (Red) c and d are on the outer boundary and cd connects opposite sides of τ.
3. (Light green) c lies on the inner boundary and cd connects perpendicular sides of τ.
4. (Dark green) c lies on the inner boundary and cd connects parallel sides of τ.

In Cases 1 and 3, constructing the axis-parallel path is relatively straightforward, we can follow the outer boundary, or the extension of the inner boundary and the outer boundary. In Cases 2 and 4, we prove that Lemma 12, in combination with the $K = 100$ constructed horizontal and vertical lines, is sufficient to obtain the "Z-shaped" paths in Figure 8. ◀

The final step is to observe that, since there is an xy-monotone axis-parallel path from $T_i(a)$ to $T_i(b)$ in $T_i(G_i)$, by reversing the transformation T_i, we obtain a cone restricted path from a to b in G. By Lemma 9, the length of this path is at most $(1+\varepsilon)|ab|$. Therefore, G is a $(1+\varepsilon)$-spanner.

Number of Steiner points. A Steiner vertex is inserted at each edge crossing between edges in G_i and G_j, where $1 \le i < j \le k = O(\sqrt{1/\varepsilon})$. The next lemma will help bound the number of crossings between edges in G_i and G_j. In this lemma, we will assume that ε is sufficiently small and thus k is sufficiently large, so that $\sin \frac{3\pi}{k} = \Omega(\frac{1}{k})$.

▶ **Lemma 14.** *Let $1 \le i < j \le k = O(\sqrt{1/\varepsilon})$, and e be an edge of G_i. Then G_j contains at most $O(\sqrt{1/\varepsilon})$ edges that both (i) intersect e and (ii) are at least as long as e.*

Proof. Let τ be a tile in the Balanced Box Decomposition of $T_j(P)$. Let the outer rectangle of τ have height h and width w. Recall that T_j sends the unit vectors in the directions $i \cdot \frac{\pi}{k}$ and $(i+3) \cdot \frac{\pi}{k}$ to the unit x- and y-vectors. Therefore, the inverse transformation $T_j^{-1}(\tau)$ sends the axis-parallel rectangle τ to a parallelogram. The side lengths are preserved, but the angle between the sides is not. Specifically, $T_j^{-1}(\tau)$ is a parallelogram with side lengths h and w, and two of the four angles of the parallelogram are $\frac{3\pi}{k}$. The area of the parallelogram is $hw \cdot \sin \frac{3\pi}{k}$.

Next, we will bound the number of tiles τ where $T_j^{-1}(\tau)$ intersects e, and where one of the sides of τ is at least as long as e. Let this set of tiles be J. Each tile $\tau \in J$ has aspect ratio at most three, so both sides of τ must be at least a third of the length of e. Consider the disk centered at the midpoint of e, with radius twice the length of e. The area of this disk is $4\pi|e|^2$. Each parallelogram $T_j^{-1}(\tau)$, where $\tau \in J$ intersects e, has side lengths $\ge \frac{|e|}{3}$, and has a smallest angle equal to $\sin \frac{3\pi}{k}$. So each parallelogram $T_j^{-1}(\tau)$ covers a region of area $\Omega(|e|^2/k)$ in the interior of the ball, since $\sin \frac{3\pi}{k} = \Omega(\frac{1}{k})$ for $k \ge 1$. Moreover, the parallelograms $T_j^{-1}(\tau)$ cover disjoint regions since the tiles $\tau \in J$ are disjoint. Therefore, there are $O(k)$ tiles in J.

Finally, each edge of G_j that intersects e and are at least as long as e must lie inside some tile in J. Moreover, every tile in J contains only $O(K^2) = O(1)$ edges. Therefore, since there are $O(k)$ tiles satisfying the desired property, there are also $O(k) = O(\sqrt{1/\varepsilon})$ edges satisfying the desired property. ◀

For every edge e in G_i, there are $O(\sqrt{1/\varepsilon})$ edges in G_j that are longer than e and cross e. So there are $O(1/\varepsilon)$ edges in $\bigcup_j G_j$ longer than e and crossing e. Next, we count the total number of edge crossings between G_i and G_j for $1 \le i, j \le k$. We charge each edge crossing to the shorter edge. There are $O(n/\sqrt{\varepsilon})$ possible choices for the shorter edge, and $O(1/\varepsilon)$ possible choices for the longer edge. Therefore, $\bigcup_i G_i$ has $O(n/\varepsilon^{3/2})$ edge crossings and our noncrossing Steiner $(1+\varepsilon)$-spanner has the same number of Steiner points.

Running time analysis. Computing the Balanced Box Decomposition for n points in the plane takes $O(n \log n)$ time [11]. Each tile can be subdivided into $O(1)$ rectangles in $O(1)$ time. So each G_i can be computed in $O(n \log n)$ time. All G_i's can be computed in $O((n \log n)/\sqrt{\varepsilon})$ time. Finally, we compute the edge crossings and thus the Steiner points. All crossings

among m segments can be computed in $O(m \log m + s)$ time, where s is the number of crossings [13, 32]; also see [14, Chapter 2]. With $m = O(n/\sqrt{\varepsilon})$ and $s = O(n/\varepsilon^{3/2})$, the overall running time is bounded by $O((n \log n)/\varepsilon^{3/2})$.

Putting this all together, we obtain the following theorem.

▶ **Theorem 3.** *For every $\varepsilon > 0$ and every set of n points in Euclidean plane, there is a noncrossing Steiner $(1+\varepsilon)$-spanner with $O(n/\varepsilon^{3/2})$ Steiner vertices. Furthermore, there is such a spanner that is cone-restricted, and can be computed in $O((n \log n)/\varepsilon^{3/2})$ time.*

4 Conclusion and Future Directions

We studied Problem 1 on the minimum number of Steiner points required to construct a noncrossing Euclidean $(1+\varepsilon)$-spanner, thereby almost resolving Open Problem 17 from the survey of Bose and Smid [27]. Our lower and upper bounds, $\Omega_\mu(n/\varepsilon^{3/2-\mu}) \le s(n,\varepsilon) \le O(n/\varepsilon^{3/2})$, match up to a subpolynomial factor in ε.

An important direction for future work is to understand the trade-off between planarity, *total weight*, and the number of Steiner points. In particular, known constructions have not been analyzed in terms of lightness, and it remains open whether one can achieve near-optimal weight while retaining asymptotically optimal sparsity in noncrossing Steiner $(1+\varepsilon)$-spanners.

As discussed in Section 1, the number of *edge crossings* is another structural measure of complexity for geometric (*non-Steiner*) spanners. It remains an open problem whether a tighter analysis of the number of crossings in a (greedy) $(1+\varepsilon)$-spanner could potentially match our upper bound $O(n/\varepsilon^{3/2})$: Determine the minimum $c(n,\varepsilon)$ such that every set of n points in the plane admits a (non-Steiner) $(1+\varepsilon)$-spanner with at most $c(n,\varepsilon)$ edge crossings.

We observe that the $(1+\varepsilon)$-stretch constraint is essential for obtaining meaningful bounds. If the stretch requirement is relaxed and one merely requires that each ab-path lies inside the ellipse $\mathcal{E}_{ab}$, then for the basic example (Section 2) with $n = \Theta(\sqrt{1/\varepsilon})$ points, a grid of side length $\sqrt{\varepsilon}$ already yields a solution with $n/\sqrt{\varepsilon}$ Steiner points, but the stretch increases to $\sqrt{2}$. The approximation of straight-line segments using grid paths that satisfy additional nondegeneracy conditions is related to the literature on *digital segments* [35, 36].

Finally, the connection to disk-tube incidences raises another intriguing open problem: How many Steiner points are required to construct a noncrossing geometric graph for a set P of n points in the plane if, instead of the spanner condition, for every $a, b \in P$, one requires an ab-path whose Fréchet distance from the straight-line segment ab is at most ε?

References

1 Mohammad Ali Abam, Mark de Berg, and Mohammad Javad Rezaei Seraji. Geodesic spanners for points on a polyhedral terrain. *SIAM J. Comput.*, 48(6):1796–1810, 2019. doi:10.1137/18M119358X.

2 Eyal Ackerman and Rom Pinchasi. On the degenerate crossing number. *Discret. Comput. Geom.*, 49(3):695–702, 2013. doi:10.1007/S00454-013-9493-1.

3 Abu Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen G. Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Comput. Sci. Rev.*, 37:100253, 2020. doi:10.1016/J.COSREV.2020.100253.

4 Oswin Aichholzer, Franz Aurenhammer, Christian Icking, Rolf Klein, Elmar Langetepe, and Günter Rote. Generalized self-approaching curves. *Discret. Appl. Math.*, 109(1-2):3–24, 2001. doi:10.1016/S0166-218X(00)00233-X.

5 Oswin Aichholzer, Manuel Borrazzo, Prosenjit Bose, Jean Cardinal, Fabrizio Frati, Pat Morin, and Birgit Vogtenhuber. Drawing graphs as spanners. *Discret. Comput. Geom.*, 68(3):774–795, 2022. doi:10.1007/S00454-022-00398-5.

6 Soroush Alamdari, Timothy M. Chan, Elyot Grant, Anna Lubiw, and Vinayak Pathak. Self-approaching graphs. In *Proc. 20th Symposium on Graph Drawing (GD)*, volume 7704 of *LNCS*, pages 260–271. Springer, 2012. doi:10.1007/978-3-642-36763-2_23.

7 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993. doi:10.1007/BF02189308.

8 Patrizio Angelini, Enrico Colasante, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani. Monotone drawings of graphs. *J. Graph Algorithms Appl.*, 16(1):5–35, 2012. doi:10.7155/JGAA.00249.

9 Srinivasa Rao Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel H. M. Smid, and Christos D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In *Proc. 4th European Symposium on Algorithms (ESA)*, volume 1136 of *LNCS*, pages 514–528. Springer, 1996. doi:10.1007/3-540-61680-2_79.

10 Sunil Arya, Gautam Das, David M. Mount, Jeffrey S. Salowe, and Michiel Smid. Euclidean spanners: Short, thin, and lanky. In *Proc. 27th ACM Symposium on Theory of Computing (STOC)*, pages 489–498, 1995. doi:10.1145/225058.225191.

11 Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998. doi:10.1145/293347.293348.

12 Davood Bakhshesh and Mohammad Farshi. Angle-constrained spanners with angle at least $\pi/3$. *Inf. Process. Lett.*, 120:44–46, 2017. doi:10.1016/J.IPL.2017.01.002.

13 Ivan J. Balaban. An optimal algorithm for finding segments intersections. In *Proc. 11th Symposium on Computational Geometry (SoCG)*, pages 211–219. ACM Press, 1995. doi:10.1145/220279.220302.

14 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition edition, 2008. doi:10.1007/978-3-540-77974-2.

15 Sujoy Bhore, Arnold Filtser, Hadi Khodabandeh, and Csaba D. Tóth. Online spanners in metric spaces. *SIAM J. Discret. Math.*, 38(1):1030–1056, 2024. doi:10.1137/22M1534572.

16 Sujoy Bhore, Balázs Keszegh, Andrey Kupavskii, Hung Le, Alexandre Louvet, Dömötör Pálvölgyi, and Csaba D. Tóth. Spanners in planar domains via Steiner spanners and non-Steiner tree covers. In *Proc. 36th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4292–4326, 2025. doi:10.1137/1.9781611978322.145.

17 Sujoy Bhore and Lazar Milenkovic. Light spanners with small hop-diameter. In *Proc. 52nd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 334 of *LIPIcs*, pages 30:1–30:16. Schloss Dagstuhl, 2025. doi:10.4230/LIPIcs.ICALP.2025.30.

18 Sujoy Bhore and Csaba D. Tóth. Euclidean Steiner spanners: Light and sparse. *SIAM J. Discret. Math.*, 36(3):2411–2444, 2022. doi:10.1137/22M1502707.

19 Sujoy Bhore and Csaba D. Tóth. Online Euclidean spanners. *ACM Trans. Algorithms*, 21(1):5:1–5:22, 2025. doi:10.1145/3681790.

20 Ahmad Biniaz, Prosenjit Bose, Jean-Lou De Carufel, Cyril Gavoille, Anil Maheshwari, and Michiel H. M. Smid. Towards plane spanners of degree 3. *J. Comput. Geom.*, 8(1):11–31, 2017. doi:10.20382/JOCG.V8I1A2.

21 Nicolas Bonichon, Prosenjit Bose, Paz Carmi, Irina Kostitsyna, Anna Lubiw, and Sander Verdonschot. Gabriel triangulations and angle-monotone graphs: Local routing and recognition. In *Proc. 24th Symposium on Graph Drawing and Network Visualization (GD)*, volume 9801 of *LNCS*, pages 519–531. Springer, 2016. doi:10.1007/978-3-319-50106-2_40.

22 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and David Ilcinkas. Connections between Theta-graphs, Delaunay triangulations, and orthogonal surfaces. In *Proc. 36th Workshop on Graph Theoretic Concepts in Computer Science (WG)*, volume 6410 of *LNCS*, pages 266–278, 2010. doi:10.1007/978-3-642-16926-7_25.

23 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perković. The stretch factor of l_1-and l_∞-Delaunay triangulations. In *European Symposium on Algorithms*, pages 205–216, 2012. doi:10.1007/978-3-642-33090-2_19.

24 Nicolas Bonichon, Iyad A. Kanj, Ljubomir Perkovic, and Ge Xia. There are plane spanners of degree 4 and moderate stretch factor. *Discret. Comput. Geom.*, 53(3):514–546, 2015. doi:10.1007/S00454-015-9676-Z.

25 Prosenjit Bose, Mirela Damian, Karim Douïeb, Joseph O'rourke, Ben Seamone, Michiel Smid, and Stefanie Wuhrer. $\pi/2$-angle Yao graphs are spanners. *International Journal of Computational Geometry & Applications*, 22(1):61–82, 2012. doi:10.1142/S0218195912600047.

26 Prosenjit Bose, Darryl Hill, and Michiel Smid. Improved spanning ratio for low degree plane spanners. *Algorithmica*, 80(3):935–976, 2018. doi:10.1007/S00453-017-0305-5.

27 Prosenjit Bose and Michiel H. M. Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom.*, 46(7):818–830, 2013. doi:10.1016/J.COMGEO.2013.04.002.

28 Jean Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, 1985. doi:10.1007/BF02776078.

29 Paz Carmi and Michiel H. M. Smid. An optimal algorithm for computing angle-constrained spanners. *J. Comput. Geom.*, 3(1):196–221, 2012. doi:10.20382/JOCG.V3I1A10.

30 Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New sparseness results on graph spanners. In *Proc. 8th ACM Symposium on Computational Geometry (SoCG)*, 1992. doi:10.1145/142675.142717.

31 Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New sparseness results on graph spanners. *Int. J. Comput. Geom. Appl.*, 5:125–144, 1995. doi:10.1142/S0218195995000088.

32 Bernard Chazelle and Herbert Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39(1):1–54, 1992. doi:10.1145/147508.147511.

33 L. Paul Chew. There is a planar graph almost as good as the complete graph. In *Proc. 2nd Symposium on Computational Geometry*, pages 169–177. ACM Press, 1986. doi:10.1145/10515.10534.

34 L. Paul Chew. There are planar graphs almost as good as the complete graph. *J. Comput. Syst. Sci.*, 39(2):205–219, 1989. doi:10.1016/0022-0000(89)90044-5.

35 Man-Kwun Chiu, Matias Korman, Martin Suderland, and Takeshi Tokuyama. Distance bounds for high dimensional consistent digital rays and 2-d partially-consistent digital rays. *Discret. Comput. Geom.*, 68(3):902–944, 2022. doi:10.1007/S00454-021-00349-6.

36 Tobias Christ, Dömötör Pálvölgyi, and Milos Stojakovic. Consistent digital line segments. *Discret. Comput. Geom.*, 47(4):691–710, 2012. doi:10.1007/S00454-012-9411-Y.

37 Kenneth Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th ACM Symposium on Theory of Computing (STOC)*, pages 56–65, 1987. doi:10.1145/28395.28402.

38 Alex Cohen, Cosmin Pohoata, and Dmitrii Zakharov. A new upper bound for the Heilbronn triangle problem. Preprint, 2023. arXiv:2305.18253.

39 Alex Cohen, Cosmin Pohoata, and Dmitrii Zakharov. Lower bounds for incidences. *Inventiones Mathematicae*, 240:1045–1118, 2025. doi:10.1007/s00222-025-01331-2.

40 Hallard T. Croft, Kenneth J. Falconer, and Richard K. Guy. *Unsolved Problems in Geometry*. Problem Books in Mathematics. Springer, 1991. doi:10.1007/978-1-4612-0963-8.

41 Gautam Das, Paul Heffernan, and Giri Narasimhan. Optimally sparse spanners in 3-dimensional Euclidean space. In *Proc. 9th Symposium on Computational Geometry (SoCG)*, pages 53–62. ACM Press, 1993. doi:10.1145/160985.160998.

42 Gautam Das, Giri Narasimhan, and Jeffrey S. Salowe. A new way to weigh malnourished Euclidean graphs. In *Proc. 6th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 215–222, 1995. URL: https://dl.acm.org/doi/10.5555/313651.313697.

43 Hooman Reisi Dehkordi, Fabrizio Frati, and Joachim Gudmundsson. Increasing-chord graphs on point sets. *J. Graph Algorithms Appl.*, 19(2):761–778, 2015. doi:10.7155/JGAA.00348.

44 Ciprian Demeter and Hong Wang. Szemerédi-Trotter bounds for tubes and applications. *Ars Inveniendi Analytica*, 1:1–46, 2025. doi:10.15781/mt54-gc31.

45 Yefim Dinitz, Michael Elkin, and Shay Solomon. Low-light trees, and tight lower bounds for Euclidean spanners. *Discrete & Computational Geometry*, 43:736–783, 2010. doi:10.1007/S00454-009-9230-Y.

46 Adrian Dumitrescu and Anirban Ghosh. Lower bounds on the dilation of plane spanners. *Int. J. Comput. Geom. Appl.*, 26(2):89–110, 2016. doi:10.1142/S0218195916500059.

47 Michael Elkin and Shay Solomon. Optimal euclidean spanners: Really short, thin, and lanky. *J. ACM*, 62(5):35:1–35:45, 2015. doi:10.1145/2819008.

48 Michael Elkin and Shay Solomon. Steiner shallow-light trees are exponentially lighter than spanning ones. *SIAM Journal on Computing*, 44(4):996–1025, 2015. doi:10.1137/13094791X.

49 David Eppstein and Hadi Khodabandeh. On the edge crossings of the greedy spanner. In *Proc. 37th Symposium on Computational Geometry (SoCG)*, volume 189 of *LIPIcs*, pages 33:1–33:17. Schloss Dagstuhl, 2021. doi:10.4230/LIPIcs.SOCG.2021.33.

50 Paul Erdős. Problems and results in combinatorial geometry. In *Discrete Geometry and Convexity (New York, 1982)*, volume 440 of *Ann. New York Acad. Sci.*, pages 1–11. Wiley Online Library, 1985.

51 Stefan Felsner, Alexander Igamberdiev, Philipp Kindermann, Boris Klemz, Tamara Mchedlidze, and Manfred Scheucher. Strongly monotone drawings of planar graphs. In *Proc. 32nd Symposium on Computational Geometry (SoCG)*, volume 51 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl, 2016. doi:10.4230/LIPIcs.SOCG.2016.37.

52 Yuqiu Fu, Shengwen Gan, and Kevin Ren. An incidence estimate and a Furstenberg type estimate for tubes in $\mathbb{R}^2$. *Journal of Fourier Analysis and Applications*, 28:59:1–28, 2022. doi:10.1007/s00041-022-09953-3.

53 Larry Guth, Noam Solomon, and Hong Wang. Incidence estimates for well spaced tubes. *Geometric and Functional Analysis*, 29:1844–1863, 2019. doi:10.1007/s00039-019-00519-y.

54 Bardia Hamedmohseni, Zahed Rahmati, and Debajyoti Mondal. Emanation graph: A plane geometric spanner with Steiner points. *Graphs Comb.*, 39(2):38, 2023. doi:10.1007/S00373-023-02632-0.

55 Christian Icking, Rolf Klain, and Elmar Langetepe. Self-approaching curves. *Mathematical Proceedings of the Cambridge Philosophical Society*, 125(3):441–453, 1999. doi:10.1017/S0305004198003016.

56 Iyad A. Kanj, Ljubomir Perkovic, and Duru Türkoglu. Degree four plane spanners: Simpler and better. *J. Comput. Geom.*, 8(2):3–31, 2017. doi:10.20382/JOCG.V8I2A2.

57 J. Mark Keil. Approximating the complete Euclidean graph. In *Proc. 1st Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 318 of *LNCS*, pages 208–213. Springer, 1988. doi:10.1007/3-540-19487-8_23.

58 J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7:13–28, 1992. doi:10.1007/BF02187821.

59 Philipp Kindermann, André Schulz, Joachim Spoerhase, and Alexander Wolff. On monotone drawings of trees. In *Proc. 22nd International Symposium on Graph Drawing (GD)*, volume 8871 of *LNCS*, pages 488–500. Springer, 2014. doi:10.1007/978-3-662-45803-7_41.

60 János Komlós, János Pintz, and Endre Szemerédi. A lower bound for Heilbronn's problem. *Journal of the London Mathematical Society*, 2-25:13–24, 1982. doi:10.1112/jlms/s2-25.1.13.

61 Hung Le, Lazar Milenkovic, and Shay Solomon. Sparse Euclidean spanners with optimal diameter: A general and robust lower bound via a concave inverse-ackermann function. In *Proc. 39th International Symposium on Computational Geometry (SoCG)*, volume 258 of *LIPIcs*, pages 47:1–47:17, 2023. doi:10.4230/LIPIcs.SOCG.2023.47.

62 Hung Le and Shay Solomon. A unified framework for light spanners. In *Proc. 55th ACM Symposium on Theory of Computing (STOC)*, pages 295–308, 2023. doi:10.1145/3564246.3585185.

63 Hung Le and Shay Solomon. Truly optimal Euclidean spanners. *SIAM J. Comput.*, 54(4):S19–135, 2025. doi:10.1137/20M1317906.

64 Hung Le, Shay Solomon, Cuong Than, Csaba D. Tóth, and Tianyi Zhang. Towards instance-optimal euclidean spanners. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 1579–1609. IEEE, 2024. doi:10.1109/FOCS61266.2024.00099.

65 Keenan Lee and André van Renssen. Generalized sweeping line spanners. *Theoretical Computer Science*, 989:114390, 2024. doi:10.1016/J.TCS.2024.114390.

66 Tom Leighton and Ankur Moitra. Some results on greedy embeddings in metric spaces. *Discret. Comput. Geom.*, 44(3):686–705, 2010. doi:10.1007/S00454-009-9227-6.

67 Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995. doi:10.1007/BF01200757.

68 Anna Lubiw and Debajyoti Mondal. Construction and local routing for angle-monotone graphs. *J. Graph Algorithms Appl.*, 23(2):345–369, 2019. doi:10.7155/JGAA.00494.

69 Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007. doi:10.1017/cbo9780511546884.

70 János Pach and Géza Tóth. Degenerate crossing numbers. *Discret. Comput. Geom.*, 41(3):376–384, 2009. doi:10.1007/S00454-009-9141-Y.

71 Christos H. Papadimitriou and David Ratajczak. On a conjecture related to geometric routing. In *Proc. 1st Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, volume 3121 of *LNCS*, pages 9–17. Springer, 2004. doi:10.1007/978-3-540-27820-7_3.

72 Satish B. Rao and Warren D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proc. 13th ACM Symposium on Theory of Computing (STOC)*, pages 540–550, 1998. doi:10.1145/276698.276868.

73 Kevin Ren and Hong Wang. Furstenberg sets estimate in the plane. Preprint, 2025. arXiv:2308.08819.

74 Günter Rote. Curves with increasing chords. *Mathematical Proceedings of the Cambridge Philosophical Society*, 115(1):1–12, 1994. doi:10.1017/S0305004100071875.

75 Klaus F. Roth. On a problem of Heilbronn. *Journal of the London Mathematical Society*, 26(3):198–204, 1951. doi:10.1112/jlms/s1-26.3.198.

76 Jim Ruppert and Raimund Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proc. 3rd Canadian Conference on Computational Geometry (CCCG)*, pages 207–210, 1991.

77 Shay Solomon. Euclidean Steiner shallow-light trees. *J. Comput. Geom.*, 6(2):113–139, 2015. doi:10.20382/jocg.v6i2a7.

78 Endre Szemerédi and William T. Trotter. Extremal problems in discrete geometry. *Combinatorica*, 3(3-4):381–392, 1983. doi:10.1007/BF02579194.

79 André van Renssen and Gladys Wong. Bounded-degree spanners in the presence of polygonal obstacle. *Theor. Comput. Sci.*, 854:159–173, 2021. doi:10.1016/J.TCS.2020.12.024.

80 Thomas Wolff. Recent work connected with the Kakeya problem. In *Prospects in Mathematics (Princeton, NJ, 1996)*, pages 129–162. AMS, Providence, RI, 1999.

81 Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4):721–736, 1982. doi:10.1137/0211059.

82 Dmitrii Zakharov. Upper bounds for Heilbronn's triangle problem in higher dimensions. *Bull. London Math. Soc.*, 56:1687–1697, 2024. doi:10.1112/blms.13020.

Flip Distance of Non-Crossing Spanning Trees: NP-Hardness and Improved Bounds

Håvard Bakke Bjerkevik
University at Albany, SUNY, NY, USA

Joseph Dorfer
Graz University of Technology, Austria

Linda Kleist
University of Hamburg, Germany

Torsten Ueckerdt
Karlsruhe Institute of Technology, Germany

Birgit Vogtenhuber
Graz University of Technology, Austria

Abstract

We consider the problem of reconfiguring non-crossing spanning trees on point sets. For a set P of n points in general position in the plane, the flip graph $\mathcal{F}(P)$ has a vertex for each non-crossing spanning tree on P and an edge between any two spanning trees that can be transformed into each other by the exchange of a single edge (coined a flip). This flip graph has been intensively studied, lately with an emphasis on determining its diameter $\operatorname{diam}(\mathcal{F}(P))$ for sets P of n points in convex position. For this case, the current best bounds are $14/9 \cdot n - O(1) \leq \operatorname{diam}(\mathcal{F}(P)) < 15/9 \cdot n - 3$, obtained in a recent breakthrough work [Bjerkevik, Kleist, Ueckerdt, and Vogtenhuber; SODA 2025]. The crucial tool for both the upper and lower bound are so-called *conflict graphs*, which the authors stated might be the key ingredient for determining the diameter (up to lower-order terms).

In this paper, we pick up the concept of conflict graphs from the above-mentioned work and show that this tool is even more versatile than previously hoped. As our first main result, we use conflict graphs to show that computing the flip distance between two non-crossing spanning trees is NP-hard, even for point sets in convex position. Interestingly, the result still holds for more constrained flip operations, concretely, compatible flips (where the removed and the added edge do not cross) and rotations (where the removed and the added edge share an endpoint).

Additionally, we present new insights on the diameter of the flip graph, by this directly extending the line of research from [BKUV SODA25]. Their lower bound is based on a constant-size pair of trees, one of which is of a type we refer to as *stacked*. We show that if one of the trees is stacked, then the lower bound is indeed optimal up to a constant term, that is, there exists a flip sequence of length at most $14/9 \cdot (n-1)$ to any other tree.

Lastly, we improve the lower bound on the diameter of the flip graph $\mathcal{F}(P)$ for n points in convex position to $11/7 \cdot n - o(n)$.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics; Mathematics of computing → Combinatorics; Mathematics of computing → Combinatoric problems

Keywords and phrases Non-crossing, spanning tree, plane graph, flip graph, reconfiguration, diameter, complexity, NP-hard, edge exchange, compatible flip, rotation, happy edge property

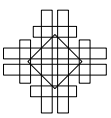
Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.16

Related Version *Full Version*: https://arxiv.org/abs/2603.22262 [12]

Funding *Joseph Dorfer*: Austrian Science Fund (FWF) 10.55776/DOC183.
Torsten Ueckerdt: funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 520708409.
Birgit Vogtenhuber: Austrian Science Fund (FWF) 10.55776/DOC183.



© Håvard Bakke Bjerkevik, Joseph Dorfer, Linda Kleist, Torsten Ueckerdt, and Birgit Vogtenhuber;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Non-crossing configurations such as triangulations, spanning trees, matchings, or polygonizations on planar point sets are fundamental structures in computational geometry, with wide applications. In dynamic environments or interactive settings, it is often necessary to transform one configuration into another through a sequence of local modifications, while ensuring that the non-crossing property is preserved at every step and that the intermediate configurations are of the same type. This process, known as (discrete) reconfiguration, raises natural algorithmic questions: Given two non-crossing configurations, is it possible to convert one into the other using only basic operations? If so, it is of interest to bound the length of a shortest reconfiguration sequence between any two configurations, to compute the length of an optimal sequence between two given configurations, and to efficiently determine a short(est) sequence.

These questions can be nicely rephrased in terms of a so-called *flip graph*. The flip graph of a discrete reconfiguration problem has a vertex for each configuration and an edge between any two configurations that can be transformed into one another by a single basic discrete operation, called a *flip*. In terms of the flip graph, the above-mentioned three questions then read as follows: Is the flip graph connected? What is the diameter of the flip graph? Can a shortest path (flip sequence) between two vertices of the flip graph be computed efficiently?

For many types of geometric graphs, the most classical flip operation is the exchange of one edge with another edge, see also Figure 1. For instance, it is known that the flip graph of triangulations on a set of n points in the plane in convex position [31] and has diameter exactly $2n - 10$ [35, 36]. For decades, it has been a tantalizing and important open problem in theoretical computer science whether a shortest flip sequence between two triangulations of a convex point set can be computed in polynomial time. This problem is equivalent to the complexity of computing the rotation distance between two rooted binary trees.

In this work, we consider flips between non-crossing geometric spanning trees. Let P denote a set of n points in the plane in general position, that is, no three points of P are collinear. A *non-crossing spanning tree on* P is a spanning tree that has P as its vertex set and whose edges are pairwise non-crossing straight-line segments. A *flip* in a non-crossing spanning tree T on P is the exchange of one edge of T by a different one such that the resulting graph is again a non-crossing spanning tree T' on P; see Figure 1 for an example. The according flip graph $\mathcal{F}(P)$ has a vertex for each non-crossing spanning tree on P and an edge between any two trees whenever they can be transformed into each other by a single edge flip. For two trees T and T' on P, we denote their flip distance, the length of a shortest path between T and T' in $\mathcal{F}(P)$, by $\mathrm{dist}(T, T')$.

In 1996, Avis and Fukuda [10] showed in their famous reverse search paper that $\mathcal{F}(P)$ is connected for any n-point set P in general position, and that its radius is bounded from above by $n-2$, implying $\mathrm{diam}(\mathcal{F}(P)) \leq 2n-4$. In 1999, Hernando, Hurtado, Márquez, Mora,

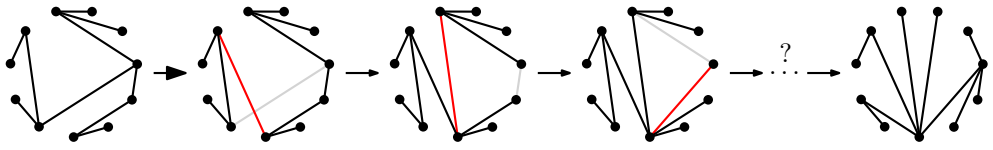


Figure 1 Reconfiguration of non-crossing spanning trees on convex point sets by edge exchange. The appearing edge is highlighted in red and the disappearing edge in gray. Can you complete the sequence to obtain the tree on the right? How many further flips are needed?

and Noy [25] showed that for convex n-point sets P, it holds that $\operatorname{diam}(\mathcal{F}(P)) \geq \lfloor 3/2 \cdot n \rfloor - 5$. Since then, the flip graph $\mathcal{F}(P)$ of non-crossing geometric spanning trees on point sets P in convex position has been subject of intensive study; see the discussion in Section 1.1 below for some details. For a long time, it was conjectured that the true value for its diameter should be $3/2 \cdot n + c$ for some constant c. Very recently, Bjerkevik, Kleist, Ueckerdt, and Vogtenhuber [13] refuted this conjecture by obtaining the currently best known bounds of $14/9 \cdot n + O(1) \leq \operatorname{diam}(\mathcal{F}(P)) \leq 15/9 \cdot n - O(1)$ for the diameter of the flip graph on non-crossing spanning trees on point sets in convex position. The crucial tool for both the upper and lower bound are so-called *conflict graphs*, which the authors introduced and stated might be the key ingredient for determining the diameter of the flip graph of spanning trees for point sets in convex position (up to lower-order terms). Their lower bound is based on a constant-size example containing a tree belonging to a family of trees that we call *stacked*.

In this work, we pick up the concept of conflict graphs from [13] and show that this tool is even more versatile than previously hoped. We prove that the flip distance between two trees can be as high as $11/7 \cdot n - o(n)$, by this directly extending the line of research from [13].

▶ **Theorem 1.** *For any set P of $n \geq 3$ points in convex position, the flip graph $\mathcal{F}(P)$ of non-crossing spanning trees on P has diameter at least $(11/7 - o(1))n$.*

We further show that if one of the trees is stacked, then the lower bound from [13] is indeed the best possible. In other words, whenever one of the trees is stacked, then there exists a flip sequence of length at most $14/9 \cdot n - O(1)$ to any other tree.

▶ **Theorem 2.** *Let T, T' be non-crossing trees on $n \geq 3$ points in convex position. If T is a stacked tree, then $\mathit{dist}(T, T') \leq \frac{14}{9}(n-1)$.*

Moreover, we use conflict graphs beyond the expectations of [13], showing that they can also be used to prove that computing the flip distance between two non-crossing spanning trees is NP-hard, even for point sets in convex position.

▶ **Theorem 3.** *Given two non-crossing spanning trees T, T' on a point set and an integer δ, the decision problem of whether there exists a flip sequence of length at most δ between T and T' is NP-complete, even for point sets in convex position.*

Interestingly, this result still holds for more constrained flip operations, concretely, *compatible* flips (where the removed and the added edge do not cross) and *rotations* (where the removed and the added edge share an endpoint). In Figure 1, while the first flip is just an edge exchange, the second flip a compatible flip, and the third flip is a rotation.

▶ **Theorem 4.** *Given two non-crossing spanning trees T, T' on a point set and an integer δ, the decision problem of whether there exists a compatible flip sequence / a rotation sequence of length at most δ between T and T' is NP-complete, even for point sets in convex position.*

We point out that the latter two statements constitute the first hardness results for flip graphs on point sets in convex position. Moreover, Theorem 4 refutes the long-held belief that the flip distance can be computed in polynomial time when the so-called happy edge property holds, because that property holds for compatible flips of non-crossing spanning trees on convex point sets. See Section 1.1 for more background and Table 1 for an overview of complexity status for various flip graphs of non-crossing structures on point sets.

Outline. We discuss further related work in Section 1.1 and introduce the central concepts from [13] that are required for our work in Section 2. In Section 3, we prove NP-completeness for deciding the length of a shortest (general, compatible, or rotational) flip sequence between

Table 1 Overview of complexity results. Results obtained in this work are marked by [*].

setting	flip type	convex position	general position
triangulations	edge exchange	NP-hard? [20][1]	NP-hard [32, 34]
non-crossing spanning trees	edge exchange	NP-hard [*]	NP-hard [*]
	compatible	NP-hard [*]	NP-hard [*]
	rotation	NP-hard [*]	NP-hard [*]
	slide	open	open
non-crossing spanning paths	edge exchange	in P [4]	open
non-crossing perfect matchings	edge exchange	in P [26]	NP-hard [11]
non-crossing odd matchings	edge exchange	open	NP-hard [3]

two spanning trees on a convex point set. In Section 4, we show that the flip distance between two trees on a convex n-point set is at most $14/9 \cdot (n-1)$ if one of them is stacked. In Section 5, we improve the lower bound on the diameter of the tree flip graph for n points in convex position to $(11/7 - o(1))n$. We conclude with a summary and open problems in Section 6. Full details and proofs of all statements marked by ($\star$) can be found in the arXiv version of our paper [12].

1.1 Related work

We review the related work in two aspects, concerning the connectedness and diameter bounds on the one hand, and the computational complexity of computing a shortest flip sequence on the other hand.

Connectedness and Diameter. For triangulations, connectedness of the flip graph for point sets was shown by Lawson [31] and for simple polygons by Hurtado, Noy, and Urrutia [28]. In both cases, the diameter of the flip graph is in $O(n^2)$. Hurtado, Noy, and Urrutia [28] provided a matching lower bound of $\Omega(n^2)$ for both settings. For triangulations of convex point sets, Sleator, Tarjan, and Thurston [36] showed that the diameter of the flip graph is at most $2n - 10$, which is tight for $n > 13$ as proven by Pournin [35]. There are many more results on flips in triangulations; see for example [21–24, 27, 37–39].

For non-crossing spanning trees of point sets in general position, Avis and Fukuda [10] showed that the flip graph is connected for flips, compatible flips, and rotations, and has a diameter at most $2n - 4$. Nichols, Pilz, Tóth, and Zehmakan [33] gave an upper bound on the diameter of $O(n \log(n))$ for empty triangle rotations. Aicholzer, Aurenhammer, and Hurtado [1] proved that the flip graph is connected for edge slides (where the source and the target edge together with some other edge of the tree form an empty triangle). They also provided an exponential upper bound on the diameter, which was later improved to $O(n^2)$ by Aichholzer and Reinhart [8]. For point sets in convex position, determining the exact diameter of the flip graph has gained considerable attention in the last couple of years. Starting from the benchmark upper bound of $2n - 4$ [10] and the conjectured to be essentially tight lower bound of $\lfloor 3n/2 \rfloor - 5$ [25], the upper bound was subsequently improved

[1] In a very recent arXiv preprint, Dorfer [20] announced that computing the flip distance between triangulations of convex point sets is NP-complete. This (potential since not yet reviewed) result was inspired by and initiated after the submission of the work at hand, and uses similar techniques: Introducing a notion of conflict graphs, showing that the flip distance is closely tied to the size of a largest acyclic subset, and proving NP-hardness of computing the largest acyclic subset.

to $2n - \log(n)$ [2] and $2n - \sqrt{n}$ [16]. The leading constant factor of 2 was broken by Bousquet, de Meyer, Pierron, and Wesolek [15], who improved the upper bound to $1.96n$. The latest advancement before the work at hand is due to the recent work [13], which improved the upper bound to $5/3 \cdot n - 3$ and the lower bound to $14/9 \cdot n - O(1)$. We note that the lower bound of $14/9 \cdot n - O(1)$ is not only the best bound for points in general position, but also holds for restricted flip types. When restricting the flips to compatible flips or rotations in the convex setting, the best standing upper bounds on the diameter of the flip graphs of spanning trees are $5/3 \cdot n - 2$ and $7/4 \cdot (n-1)$, respectively, due to Aichholzer, Dorfer, and Vogtenhuber [5].

For plane spanning paths, connectedness of the flip graph is an interesting open problem. So far, the flip graph has been shown to be connected for convex point sets [9] (with diameter $2n - 6$ for $n > 4$ [18]), wheel sets and double circles [6], as well as point sets with two convex layers [30]. For matchings, the connectedness problem becomes a parity issue: While the flip graph of odd matchings is known to be connected [3], the connectedness of the flip graph remains an open question for perfect matchings. Partial progress has been made for convex point sets by Hernando, Hurtado, Noy [26] and for the case when an unbounded number of edges can be exchanged by Houle, Hurtado, Noy, and Rivera-Campo [27].

Computational complexity of shortest flip sequences and the happy edge property. In several settings, there is a close connection between the existence of NP-hardness proofs or polynomial-time algorithms and the happy edge property. A reconfiguration problem has the *happy edge property* if edges that are in the initial as well as the target configuration (and thus are *happy*) can remain in the configuration throughout the entire flip sequence.

The happy edge property holds for triangulations of point sets in convex position [36], but not for triangulations of general point sets or polygons [14]. The latter fact was used to show that the flip distance problem for triangulations of point sets and polygons is NP-complete [7,32] and also APX-hard on point sets [34]. In contrast, on point sets without empty convex pentagons, the flip distance can be computed in polynomial time [21]. If we consider point sets in convex position, the flip graph is the 1-skeleton of a famous polytope, the *associahedron*. For the more general case of graph associahedra [19], the shortest path problem has been recently shown to be NP-hard [29]. Finding shortest paths in the associahedron or equivalently shortest flip sequences between triangulations of convex point sets might also be hard by a very recent arXiv preprint [20].

In the case of perfect matchings on general point sets, NP-hardness of computing a shortest flip sequence was shown via counterexamples to the happy edge property [11]. For convex point sets, the happy edge property holds and the flip distance between two matchings can be determined in linear time [26]. Similarly, for the reconfiguration of odd matchings on general point sets, the happy edge property also does not hold and the flip distance is NP-hard to compute [3].

All these results support the long standing belief that the happy edge property is tied to the complexity of graph reconfiguration problems: If the happy edge property holds, then the problem is supposedly easy to solve. If it doesn't hold, the problem should be hard to solve.

Most recently, this pattern has been broken, when Aichholzer and Dorfer [4] provided a linear time algorithm to determine the flip distance between non-crossing spanning paths of convex point sets, despite the existence of counterexamples to the happy edge property. What remained open (until now) is the other direction: Is there a graph reconfiguration problem for which it is NP-hard to compute the flip distance, even though the problem has the happy edge property? We give an affirmative answer to this question in this work: The happy edge property holds for compatible flips on non-crossing spanning trees on convex point sets [5], but the flip distance is NP-hard to compute as shown in Theorem 4.

2 Fundamental concepts: Edge pairs, conflict graphs, and blowups

Our results build on insights from Bjerkevik, Kleist, Ueckerdt, and Vogtenhuber [13]. In the following, we introduce some of their concepts.

Linear Representation. A convex point set is usually visualized by equally spaced points on a circle. A *linear representation* of a non-crossing tree T on a convex point set is constructed by cutting open the circle and unfolding the points into a horizontal line segment, called the spine. Edges in T can be illustrated nicely by semi-circles connecting their end points above the spine. Given two trees (for example the initial tree and the target tree of a flip sequence) in the same linear representation, it is convenient to represent the edges of one tree above the spine and the edges of the other tree below the spine. For an example, see Figure 2. If an edge between two consecutive vertices on the spine is contained in both trees we will in later parts of the paper depict this by drawing it as a straight line between the two vertices instead of two semicircles.

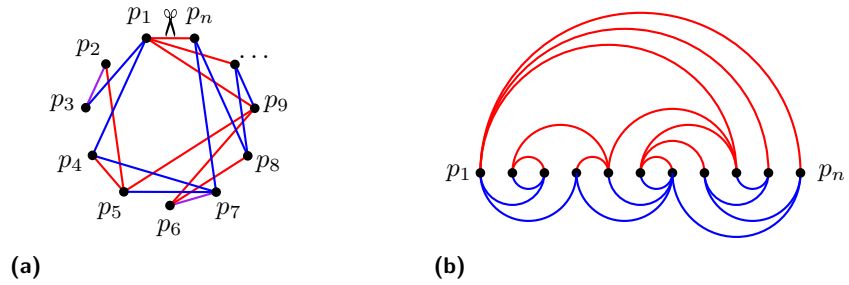


Figure 2 Figure reproduced from [13]. a Two non-crossing trees T, T' on a circularly labeled point set in convex position and b its linear representation with T above and T' below the spine.

By labeling the vertices along the linear representation by $p_1, ..., p_n$, we get a natural notion of the *length* of an edge. Concretely, the length of the edge p_ip_j is defined as $|i - j|$. We say a point p_k is *covered* by an edge p_ip_j with $i < j$ if $i \leq k \leq j$. An edge *covers* another edge if it covers both of its endpoints.

Gaps and Edge Pairing. A *gap* in the convex hull is the open line segment between two consecutive points p_i and p_{i+1} on the spine. For every tree T, there is a natural *gap-edge bijection* between gaps in the linear representation and edges of T: Each gap is assigned to the shortest edge that covers the gap. This assignment is indeed a bijection by Lemma 3.1 in [13]. For an example, consider Figure 3a. Based on the assigned gap, edges of T are partitioned into three groups. In particular, an edge is *short* if it shares both vertices with its gap, *near* if it shares one vertex with its gap, and *long* if shares no vertex with its gap.

For two trees T and T', we obtain an edge-edge bijection by pairing two edges that are assigned to the same gap of the linear representation. The authors of [13] explain how to handle nine types of edge pairs. For our results it is sufficient to concentrate on near-near pairs where the two edges are distinct. We call a gap g_i with edges e_i from T and e'_i from T'

above if e_i and e'_i are adjacent and e_i is longer than e'_i, see Figure 3b.

below if e_i and e'_i are adjacent and e_i is shorter than e'_i, see Figure 3c.

crossing if e_i and e'_i are not adjacent, equivalently, the two edges cross if drawn in the same convex point set, see Figure 3d.

The set of all above, below, and crossing gaps is denoted by A, B, and C, respectively.

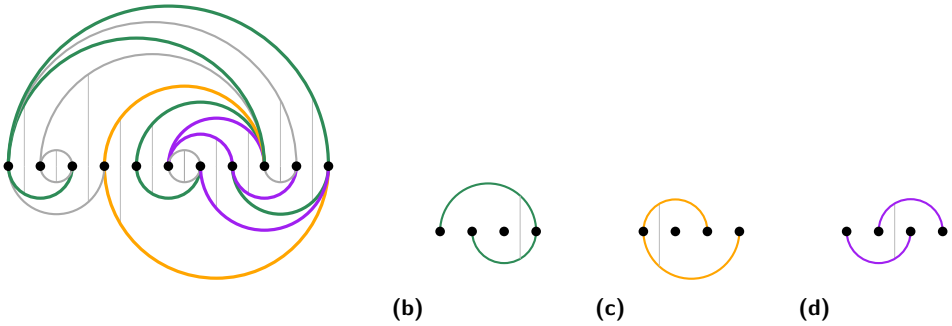


Figure 3 a Illustration of the gap-edge and edge-edge bijection. Examples of an above b, a below c, and a crossing pair d. Edges belonging to near-near pairs in A, B, and C are colored green, orange, and purple, respectively (also in several other figures).

Conflict Graphs. *Conflict graphs* were introduced as a tool to find the largest set of near-near pairs where the initial edge can be flipped directly to the target edge. The conflict graph H has a vertex for each (gap associated to a) near-near pair and a directed edge from a pair (e_i, e_i') to (e_j, e_j') if the edge e_i needs to be removed before the flip that adds e_j' and removes e_j can be performed. We denote the set of vertices in H by $V(H)$. There exist three types of conflicts. Specifically, there is a directed edge $\overrightarrow{g_i g_j}$ in H, from g_i to g_j if

type 1 e_i crosses e_j', see Figure 4a.

type 2 e_j' covers e_i and e_i covers g_j, see Figure 4b.

type 3 e_i covers e_j' and e_j' covers g_i, see Figure 4c.

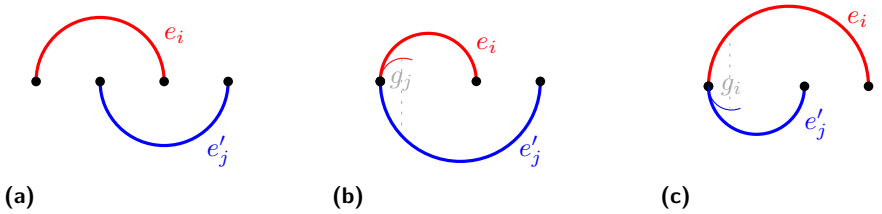


Figure 4 Figure reproduced from [13]. Examples of conflicts: a type 1, b type 2, c type 3.

We slightly abuse terminology and view the vertices of H interchangeably as gaps and as near-near pairs. Let $\mathrm{ac}(H)$ denote the largest number of vertices in H which induce an acyclic subgraph. The authors of [13] derive the following relations between acyclic sets in the conflict graph and flip distances.

▶ **Theorem 5** ([13, Theorem 2.1]). *Let T, T' be two non-crossing trees on linearly ordered points $p_1, \ldots, p_n$ with corresponding conflict graph $H = H(T, T')$.*

(i) *If $V(H)$ is non-empty, then* $\mathrm{dist}(T, T') \leq \max\{\frac{3}{2}, 2 - \frac{\mathrm{ac}(H)}{|V(H)|}\}(n-1)\}$. *If $V(H)$ is empty, then* $\mathrm{dist}(T, T') \leq \frac{3}{2}(n-1)$.

(ii) *If $V(H)$ is non-empty, then there is a constant c depending only on T and T' such that for all $\bar{n} \geq 1$, we have* $\mathrm{diam}(\mathcal{F}_{\bar{n}}) \geq \left(2 - \frac{\mathrm{ac}(H)}{|V(H)|}\right)\bar{n} - c$.

Moreover, we will use the following lemma on the sets of above, below, and crossing gaps.

▶ **Lemma 6** ([13, Lemma 3.2]). *Each of A, B, and C is an acyclic set of H.*

Blowups. By default, a flip sequence based on the described edge-edge bijection is not necessarily a shortest one and can in fact be very far from optimal. The authors of [13] overcome this by introducing *blowups* for near-near pairs. For trees T and T', the β-blowup is a pair of trees $\beta \cdot T$ and $\beta \cdot T'$ obtained by the following construction. For every gap g that corresponds to a near-near pair (e, e'), insert a set $V(e) = V(e')$ of β additional points in the gap g. In $\beta \cdot T$ add an edge from each $v \in V(e)$ to the endpoint of e that is not adjacent to g. Proceed similarly in $\beta \cdot T'$ for e'. Part (ii) of Theorem 5 is obtained by considering blowups.

▶ **Lemma 7** ([13, Lemma 23, arXiv-Version]). *The flip distance from $\beta \cdot T$ to $\beta \cdot T'$ is at least $(\beta - 2n)(2|V(H)| - \mathrm{ac}(H))$, where $H = H(T, T')$.*

3 NP-Completeness results

In this section we sketch proofs for our hardness results. Theorem 3 states that the decision problem of whether there exists a flip sequence of length at most δ between two non-crossing spanning trees T, T' is NP-complete. In fact, this holds even for point sets in convex position. Our proof of Theorem 3 goes via an intermediate problem. Specifically, we show that it is NP-complete to decide whether a directed graph H has an acyclic subset of size at least k, even if H is the conflict graph of two non-crossing trees with a given linear representation.

▶ **Proposition 8.** *Given two non-crossing trees T, T' on a convex point set, with a linear representation and corresponding non-empty conflict graph $H = H(T, T')$, as well as an integer k, it is* NP*-complete to decide whether* $\mathrm{ac}(H) \geq k$.

Our proof of Proposition 8 is a reduction from the NP-complete problem Planar Max-2SAT. We consider a restricted version [17], in which we are given a 2SAT-formula ϕ with variable-set X and clause-set $\mathcal{C}$ whose vertex-clause incidence graph G_ϕ admits an *aligned drawing*, namely, a drawing with all variables on a horizontal line L and no edge crossing L or any other edge. In particular, every clause lies either in the upper or lower halfplane defined by L. Additionally given an integer t, Planar Max-2SAT asks whether X admits a truth assignment such that at least t clauses in $\mathcal{C}$ are satisfied.

For fixed ϕ and t, we construct two non-crossing trees T, T' with a linear representation and an integer k such that the following are equivalent:

- The conflict graph $H = H(T, T')$ admits an acyclic subset of size at least k.
- The 2SAT-formula ϕ admits a truth assignment that satisfies at least t clauses.

To construct T and T', we model the variables and clauses in ϕ with small gadgets. In doing so, we also discuss the conflicts that occur. Recall that each gap has a type: above, below, or crossing. In fact, it will be enough for us to focus on the *mixed conflicts*, i.e., those between gaps of different types. We also remark that our construction only has conflicts $\overrightarrow{g_i g_j}$ of type 1, i.e., where e_i properly crosses e'_j as illustrated in Figure 4a.

Let $x \in X$ be a variable and $d = \deg(x)$ be its degree in G_ϕ, i.e., x occurs (negated or non-negated) in exactly d clauses. The variable-gadget for x consists of $8d + 1$ pairs of edges, of which $6d$ are near-near pairs and hence their corresponding $6d$ gaps are vertices in the conflict graph $H = H(T, T')$. The gadget is illustrated in Figure 5a and the corresponding mixed conflicts in Figure 5b. The $6d$ gaps form a path P of double conflicts in the conflict graph H, i.e., every acyclic subset is an independent set in P. We partition P into two independent sets $t(x)$ and $f(x)$ by alternating the gaps in $t(x)$ and $f(x)$ along P. Later we ensure that every acyclic subset S of H with $|S| \geq k$ either contains all of $t(x)$ or all of $f(x)$, which encodes the two possible truth assignments True and False for variable x.

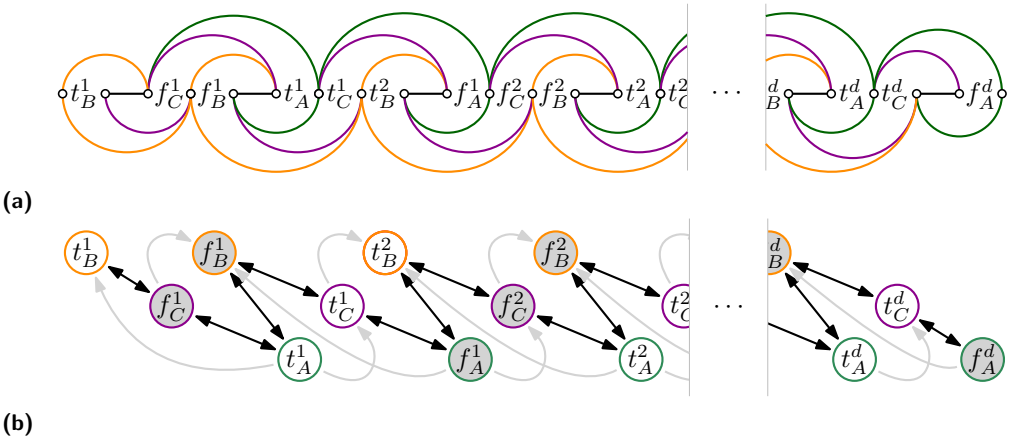


Figure 5 Illustration of a variable-gadget and the graph of its mixed conflicts.

We place each variable-gadget at the position of the corresponding variable in the aligned drawing of G_ϕ in such a way that the rightmost point of any gadget coincides with the leftmost point of the next gadget (immediately to the right). Secondly, for each clause C with incident variables x and y we introduce a new gap g_x in the variable-gadget of x and a new gap g_y in the variable-gadget of y. For each $z \in \{x, y\}$, the edge pair for g_z has an edge of length 3 in the variable-gadget of z and an edge connecting the gadgets of x and y. See Figures 6a and 7a for an illustration. To ensure that different clause-gadgets do not cross, long edges are put in the halfplane that contains the clause C in the aligned drawing of G_ϕ. Further, the position of g_z on the path P of length $6\deg(z)$ in the variable-gadget of z is chosen with respect to the order of incident edges at z in the drawing of G_ϕ.

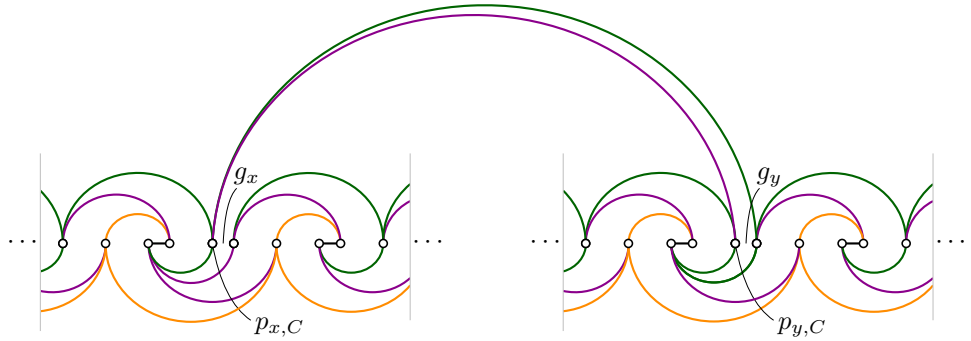
Moreover, g_z is chosen such that it has a double conflict with a gap in $t(z)$ if z appears negated in C, and a double conflict with a gap in $f(z)$ if z appears non-negated in C. This ensures that an acyclic set S contains the gap g_z only if the truth assignment of variable z satisfies the clause C. A double conflict between the two gaps g_x, g_y of the clause-gadget ensures that at most one of g_x, g_y is in any acyclic set S. There are six further mixed conflicts between g_x, g_y and the variable-gaps, as illustrated in Figures 6b and 7b. Given a truth assignment, we choose all of $t(x)$ when x is True, $f(x)$ when x is False, and one gap g_z in each satisfied clause. That this gives an acyclic subset S of $H(T, T')$, relies on the following observation.

▶ **Observation 9.** *For every mixed conflict $\overrightarrow{g_1 g_2}$ one of the following holds:*

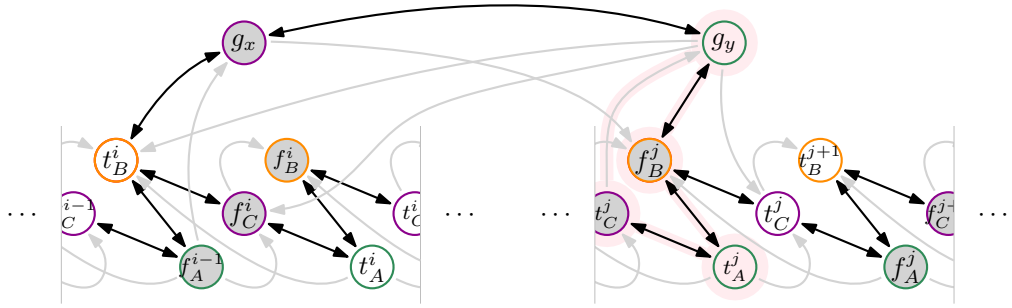
- *g_1 is above*
- *g_2 is below*
- *g_1 and g_2 have a double conflict*
- *there is a g_1-g_2-path of three double conflicts with three gaps of some variable-gadget*

In fact, each of the sets A, B, C of all above, all below, and all crossing gaps, respectively, is acyclic by Lemma 6. Thus with Observation 9 it follows that every set $S \subseteq V(H)$ with no double conflict and none of the exceptional conflicts in Observation 9 is acyclic by taking first $A \cap S$, then $C \cap S$, and finally $B \cap S$ for a topological ordering.

This way, every truth assignment satisfying t of ϕ corresponds to an acyclic set $S \subseteq V(H)$ containing exactly half of all variable-gaps and exactly t of all clause-gaps. Finally, for the other direction (getting a truth assignment from an acyclic subset S), we need to ensure that for every variable x we have $t(x) \subseteq S$ or $f(x) \subseteq S$, as long as S is large enough. To this end, let $m = |\mathcal{C}|$ and for each variable-gadget we apply an m-blowup to the first and last



(a)



(b)

Figure 6 Illustration of a clause-gadget for a clause above the horizontal line and its conflict graph. The special conflict corresponding to the last option in Observation 9 is highlighted.

gap, a $2m$-blowup to all other variable-gaps, while applying no blowup to the clause-gaps. See Figure 8 for an illustration. Setting $k = m(12m + 6 - |X|) + t$ concludes our reduction from Planar Max-2SAT and thus the proof of Proposition 8.

In order to finally prove Theorem 3, we now reduce the problem of finding large acyclic subsets in conflict graphs to the problem of finding short flip sequences in β-blowups.

▶ **Theorem 10.** *Let T and T' be two non-crossing spanning trees on n vertices with a linear representation and non-empty conflict graph $H(T, T')$, and $k > 0$ be an integer. Let $\beta = 4n^2 + 2n$. Then the following are equivalent.*

- *The conflict graph $H = H(T, T')$ has an acyclic subset of size at least k.*
- *The flip distance between $\beta \cdot T$ and $\beta \cdot T'$ is at most $(\beta+1)(2|V(H)| - k) + 2(n - |V(H)|)$.*

Proof. For convenience, let us denote $|H| = |V(H)|$. Assume H does not contain an acyclic set of size k, i.e., $\mathrm{ac}(H) < k$. Then by Lemma 7,

$$\begin{aligned}\mathrm{dist}(\beta \cdot T, \beta \cdot T') &\geq (\beta - 2n)(2|H| - \mathrm{ac}(H)) \geq (\beta + 1 - (2n + 1))(2|H| - k + 1)\\ &= (\beta + 1)(2|H| - k) + (\beta + 1) - (2n + 1)(2|H| - k + 1)\\ &> (\beta + 1)(2|H| - k) + 2(n - |H|).\end{aligned}$$

For the last inequality observe that $|H| < n$ and hence $(2n+1)(2|H|-k+1) \leq (2n+1)(2n-1) \leq 4n^2$, which together with $\beta = 4n^2 + 2n > 4n^2 + 2(n - |H|)$ gives the desired lower bound.

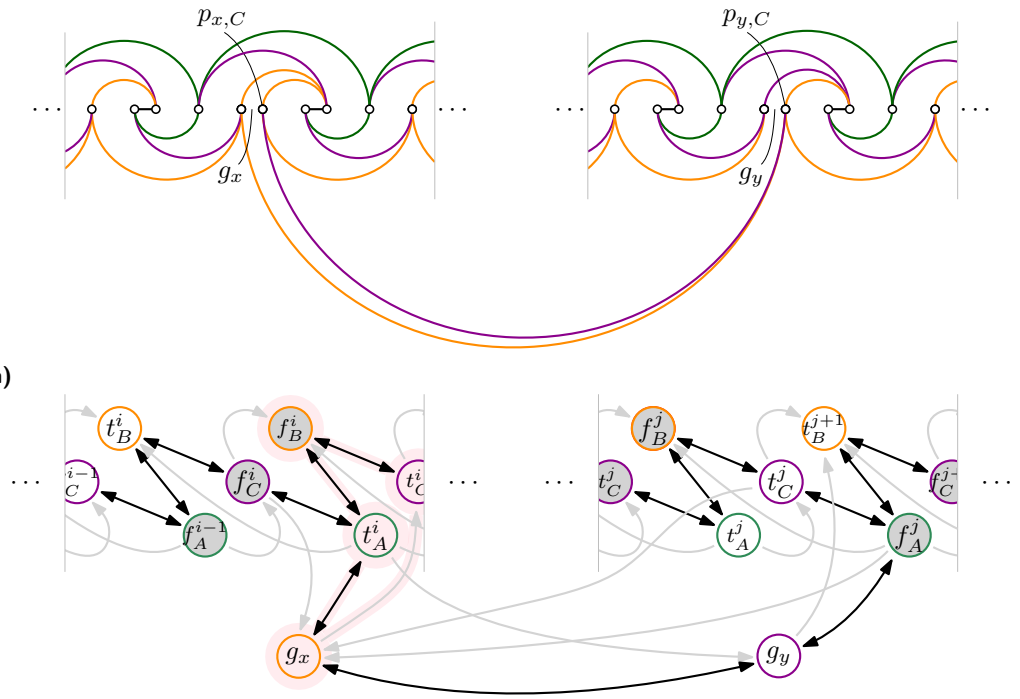


Figure 7 Illustration of a clause-gadget for a clause below the horizontal line and its conflict graph. The special conflict corresponding to the last option in Observation 9 is highlighted.

Conversely, assume $H = H(T, T')$ has an acyclic subset of size k, i.e., $\mathrm{ac}(H) \geq k$. Then, the conflict graph $H^\beta = H(\beta \cdot T, \beta \cdot T')$ of the blowup has an acyclic set of size $k \cdot (\beta + 1)$, i.e., $\mathrm{ac}(H^\beta) \geq k(\beta + 1)$. For the original trees T, T' we have n vertices and $n - 1$ gaps, of which $|H|$ are affected by the blowup. Thus, each of $\beta \cdot T$ and $\beta \cdot T'$ has $n + \beta|H|$ vertices and $n - 1 + |\beta|$ edges. Moreover, $|V(H^\beta)| = (\beta + 1)|H|$.

Using one flip for each edge pair in the largest acyclic subset and two flips for each other edge pair gives the desired bound of

$$\mathrm{dist}(\beta \cdot T, \beta \cdot T') \leq (\beta + 1)(2|H| - \mathrm{ac}(H)) + 2(n - 1 - |H|) < (\beta + 1)(2|H| - k) + 2(n - |H|).$$

and hence completes the proof. ◀

We remark that the proof of Theorem 10 produces a flip sequence that contains non-compatible flips. Next, we modify the proof to the case of flip sequences that contain only compatible flips or only rotations. In fact, a direct flip for an above pair or a below pair is already a rotation. Further, flipping any near edge to a short edge covering its associated gap is also a rotation. We do, however, need to deal with large collections of crossing edge pairs. To this end, we use the following fact.

▶ **Lemma 11** (⋆). *Let T, T' be two trees that differ in a single crossing pair. Then the rotation flip distance between $\beta \cdot T$ and $\beta \cdot T'$ is at most $\beta + 2$.*

With Lemma 11 in place, we prove that finding short compatible flip sequences and short rotation sequences is NP-hard as well. As before, we use the shorthand notation $|H| = |V(H)|$.

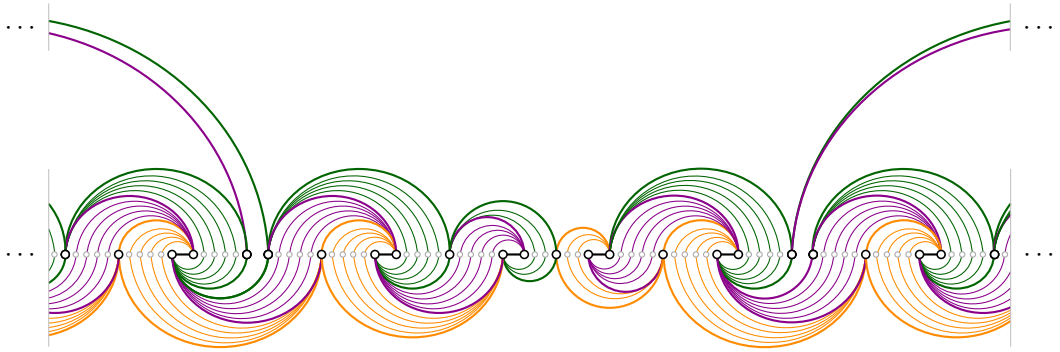


Figure 8 Applying an m-blowup (here $m = 2$) to the first and last gap of each variable-gadget and $2m$-blowups to all other gaps in variable-gadgets.

▶ **Theorem 12** ($\star$). *Let T and T' be two non-crossing spanning trees on n vertices with a linear representation, and $k > 0$ be an integer. Let $\beta = 4n^2 + 4n$. The following are equivalent.*

- *The conflict graph $H = H(T, T')$ has an acyclic subset of size at least k.*
- *The compatible flip distance* $\mathrm{dist}_{\mathrm{comp}}(\beta \cdot T, \beta \cdot T') \leq (\beta + 1)(2|H| - k) + |H| + 4(n - |H|)$.
- *The rotation flip distance* $\mathrm{dist}_{\mathrm{rot}}(\beta \cdot T, \beta \cdot T') \leq (\beta + 1)(2|H| - k) + |H| + 4(n - |H|)$.

4 Stacked Trees

In this section, we improve the upper bound for the special case where in one of the two trees, the (relevant) edges can be partitioned into independent "stacks" of nested edges. This can be seen as a first natural generalization of an upper bound where one tree is a separated caterpillar, that is, a tree with two stacks [13, Section 6, arXiv-Version]. Further, the top tree in the best previously known lower bound example in [13], depicted in Figure 9, has this property. Throughout this section, we refer to a non-crossing spanning tree on a convex point set in linear representation simply as a tree.

We call a tree T *stacked* if its edge set can be partitioned into sets $S_1, \ldots, S_k$ such that

(i) no edge in S_i covers an edge in S_j for $i \neq j$, and

(ii) each S_i is totally ordered by the covering relation.

Let T' be another tree on the same vertex set. We say that T is *stacked with respect to T'* if the subset N of edges in T that are in near-near pairs of (T, T') admits a partition into sets $S_1, \ldots, S_k$ satisfying properties (i) and (ii). Note that if T is stacked then it is stacked with respect to any T'. We refer to the edge sets S_i as *stacks (with respect to T')*.

4.1 The Special Case: Three stacks

Fix trees T and T'. We first assume that T has exactly three stacks S_1, S_2 and S_3 with respect to T'. Let $\mathcal{P}_N$ be the set of near-near pairs of (T, T'). We partition $\mathcal{P}_N$ into 9 different sets. A pair $(e, e') \in \mathcal{P}_N$ will be put into sets by the following rules.

- A_i if e belongs to stack S_i and (e, e') is an above pair.
- $B_{i,j}$, $i \neq j$ if e belongs to stack S_i, (e, e') is a below pair or a crossing pair and e' crosses an edge in a different stack S_j.
- If e belongs to stack S_i, (e, e') is a below pair or a crossing pair and e' does not cross an edge in any other stack S_j, we assign (e, e') arbitrarily to one of the two sets $B_{i,j}$.

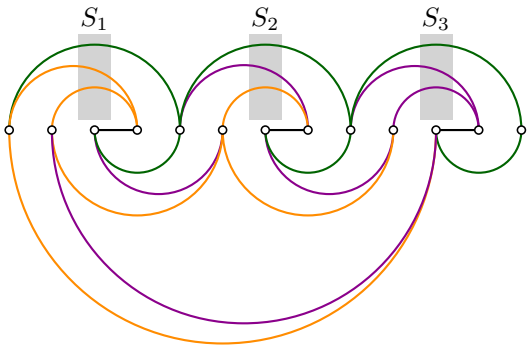


Figure 9 The best previously known lower bound example from [13]. The top tree is a stacked tree with three stacks. The bottom tree is neither stacked, nor stacked with respect to the top tree.

For this to be well-defined, we need to verify that (e, e') cannot belong to both $B_{i,j}$ and $B_{i,j'}$ for $j \neq j'$. The edge e' covers two gaps adjacent to its endpoints. One of these is covered by e. If e' intersects edges $f \in S_j$ and $g \in S_{j'}$, then the other gap is covered by both f and g, which is a contradiction, since f does not cover g or vice versa by the definition of stacks.

▶ **Lemma 13** (⋆). *For all choices of x, y, z such that $\{x, y, z\} = \{1, 2, 3\}$, both $H_x := A_y \cup A_z \cup B_{x,y} \cup B_{x,z}$ and $H_{x,y} := A_z \cup B_{y,z} \cup B_{x,y} \cup B_{x,z}$ are acyclic.*

We are now ready to obtain the following upper bound on the flip distance if one of the two trees is a stacked tree with three stacks.

▶ **Theorem 14.** *Let T, T' be non-crossing trees on $n \geq 3$ points in convex position. Let T be a stacked tree with three stacks. Then* $\mathrm{dist}(T, T') \leq {}^{14}/_{9}(n-1)$.

Proof. Each element of $V(H)$ is contained in four of the nine acyclic sets $H_1, H_2, H_3, H_{1,2}, H_{2,1}, H_{1,3}, H_{3,1}, H_{2,3}, H_{3,2}$ from Lemma 13. Thus, the average size of these sets is ${}^{1}/_{9}\Sigma_{i=1}^{9}|H_i| = {}^{4}/_{9}|V(H)|$ and, consequently, the largest acyclic set is of size at least ${}^{4}/_{9}|V(H)|$. By Theorem 5(i), there exists a flip sequence from T to T' of length at most ${}^{14}/_{9}(n-1)$. ◀

4.2 Trees with arbitrarily many stacks

We now consider a pair of trees T and T' where T is a stacked tree and let $S_1, \ldots, S_k$ denote the stacks of T with respect to T'. Let G be the graph with $S_1, \ldots, S_k$ as its vertices and an edge between S_i and S_j with $i \neq j$ if there is a near edge e in S_i paired with a near edge e' that crosses an edge in S_j. Then G is a subset of a triangulation of a convex polygon and therefore 3-colorable. We let the colors be denoted by 1, 2 and 3.

We partition the set of near-near pairs into 9 different sets, based on the 3-coloring. A pair $(e, e') \in \mathcal{P}_N$ will be put into a set by the following rules.

- A_i if e belongs to a stack with color i and (e, e') is an above pair.
- $B_{i,j}$, $i \neq j$ if e belongs to a stack with color i, (e, e') is a below pair or a crossing pair and e' crosses an edge in a stack with a color $j \neq i$.
- If e belongs to a stack with color i, (e, e') is a below or a crossing pair and e' does not cross an edge in any other stack, we assign (e, e') arbitrarily to one of the two sets $B_{i,j}$.

Lemma 15, which is a slight extension of Lemma 13, can be used to prove Theorem 16 in a similar way as Theorem 14 is derived from Lemma 13.

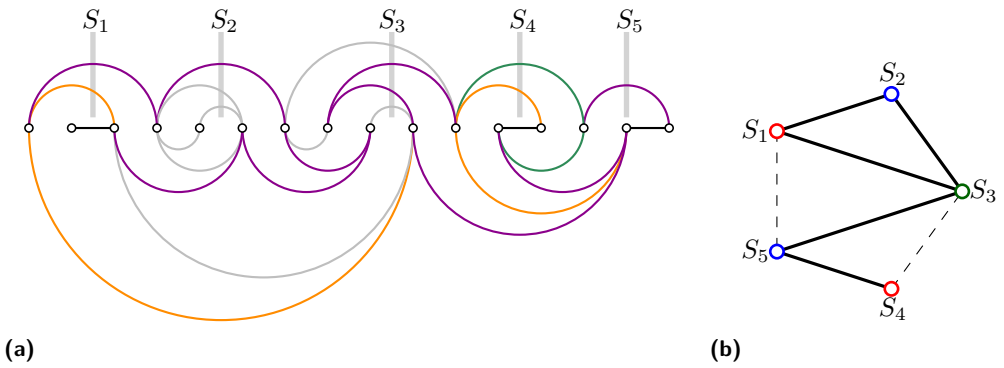


Figure 10 (a) A pair of trees where one tree is stacked and (b) the resulting graph G with a proper 3-coloring.

▶ **Lemma 15** (⋆). *Let* $\{x, y, z\} = \{1, 2, 3\}$. *Then* $H_x = A_y \cup A_z \cup B_{x,y} \cup B_{x,z}$ *and* $H_{x,y} = A_z \cup B_{y,z} \cup B_{x,y} \cup B_{x,z}$ *are acyclic.*

▶ **Theorem 16.** *Let* T, T' *be non-crossing trees on* $n \geq 3$ *points in convex position. Let* T *be stacked with respect to* T'. *Then* $\operatorname{dist}(T, T') \leq {}^{14}/_{9}(n-1)$.

We remind the reader that if T is stacked, then it is stacked with respect to any T'. Thus, the same statement without "with respect to T'" is true a fortiori.

5 A new lower bound

We construct trees T^i and $\tilde{T}^i$ iteratively, and show that their conflict graph has $7i$ vertices and a largest acyclic subset of size at most $3i + 1$. The latter divided by the former tends to ${}^{3}/_{7}$, which allows us to use Theorem 5 (ii) to prove $\operatorname{diam}(\mathcal{F}_n) \geq {}^{11}/_{7} \cdot n - o(n)$.

We start with the trees T^1 and $\tilde{T}^1$ illustrated in Figure 11. These are trees on vertices $v_1, \ldots, v_8, x, y, v_9$ labeled from left to right along the spine and have short edges v_2v_3, v_6v_7 and xy as well as edges e_i and $\tilde{e}_i$ for $i = 1, \ldots, 7$, respectively.

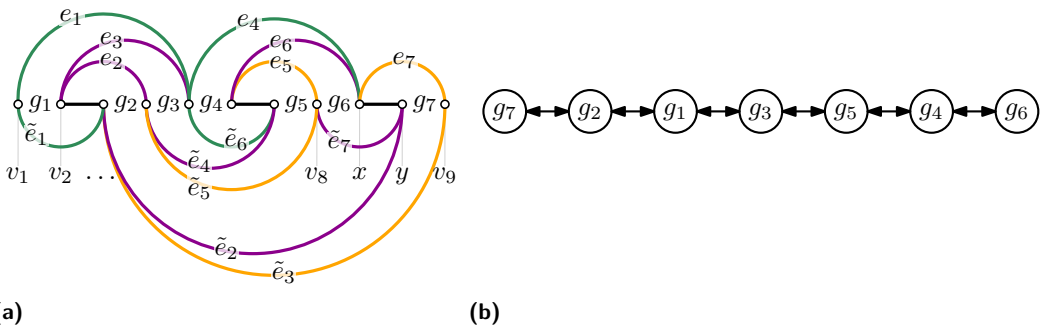


Figure 11 Illustration of T^1 and $\tilde{T}^1$ and the relevant part of the conflict graph.

Next, we construct T^ℓ and $\tilde{T}^\ell$ inductively for $\ell \geq 2$. Assume that $T^{\ell-1}$ and $\tilde{T}^{\ell-1}$ are given. Add vertices $v_1^\ell, \ldots, v_5^\ell$ from left to right to the left of $v_1^{\ell-1}$ and let $v_6^\ell = v_2^{\ell-1}$. Further, add a vertex v_7^ℓ between $v_3^{\ell-1}$ and $v_4^{\ell-1}$, a vertex v_8^ℓ between $v_8^{\ell-1}$ and x, and a vertex v_9^ℓ to the right of $v_9^{\ell-1}$. For an illustration consider Figure 12.

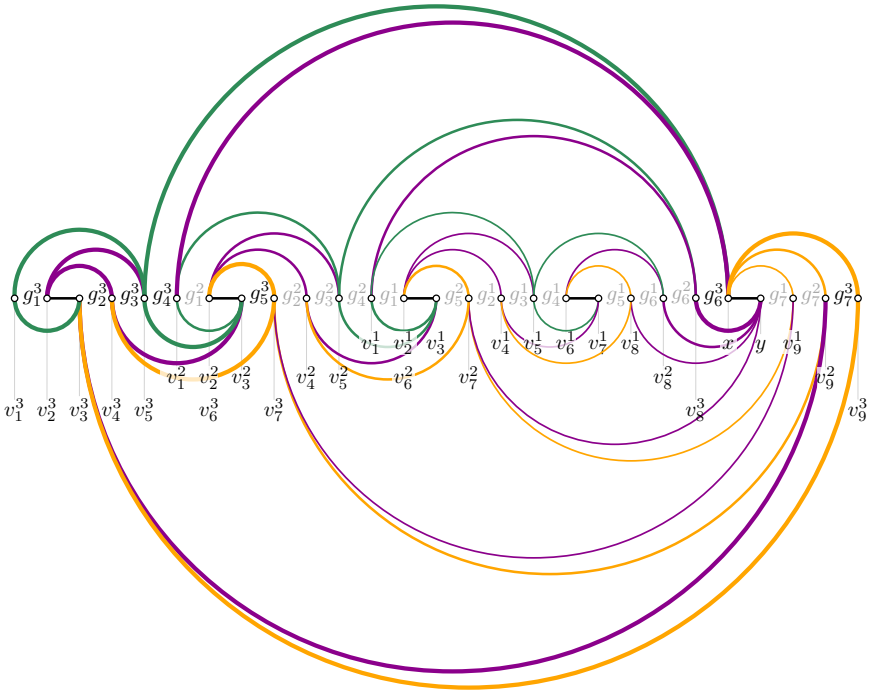


Figure 12 Illustration for the iterative construction of T^ℓ and $\tilde{T}^\ell$ from $T^{\ell-1}$ and $\tilde{T}^{\ell-1}$ for $\ell = 3$.

The edges in $T^{\ell-1}$ and $\tilde{T}^{\ell-1}$ are also present in T^ℓ and $\tilde{T}^\ell$, respectively, except that the right endpoint of $e_4^{\ell-1}$ and $e_6^{\ell-1}$ (which was x) is changed to v_8^ℓ, and the left endpoint of $\tilde{e}_2^{\ell-1}$ and $\tilde{e}_7^{\ell-1}$ (which was $v_3^{\ell-1}$) is changed to v_7^ℓ. Correspondingly, we change $g_2^{\ell-1}$ to $v_7^\ell v_4^{\ell-1}$ and $g_6^{\ell-1}$ to $v_8^{\ell-1}v_8^\ell$. We put a short edge $v_2^\ell v_3^\ell$ in both T^ℓ and $\tilde{T}^\ell$. Lastly, we add the following edges to T^ℓ and $\tilde{T}^\ell$

$$e_1^\ell = v_1^\ell v_5^\ell, \quad e_2^\ell = v_2^\ell v_4^\ell, \quad e_3^\ell = v_2^\ell v_5^\ell, \quad e_4^\ell = v_5^\ell x, \quad e_5^\ell = v_6^\ell v_7^\ell, \quad e_6^\ell = v_1^{\ell-1} x, \quad e_7^\ell = x v_9^\ell,$$
$$\tilde{e}_1^\ell = v_1^\ell v_3^\ell, \quad \tilde{e}_2^\ell = v_3^\ell v_9^{\ell-1}, \quad \tilde{e}_3^\ell = v_4^\ell v_3^{\ell-1}, \quad \tilde{e}_4^\ell = v_5^\ell v_3^{\ell-1}, \quad \tilde{e}_5^\ell = v_4^\ell v_7^\ell, \quad \tilde{e}_6^\ell = v_8^\ell y, \quad \tilde{e}_7^\ell = v_3^\ell v_9^\ell,$$

with corresponding gaps

$$g_1^\ell = v_1^\ell v_2^\ell, \quad g_2^\ell = v_3^\ell v_4^\ell, \quad g_3^\ell = v_4^\ell v_5^\ell, \quad g_4^\ell = v_5^\ell v_1^{\ell-1}, \quad g_5^\ell = v_3^{\ell-1} v_7^\ell, \quad g_6^\ell = v_8^\ell x, \quad g_7^\ell = v_9^{\ell-1} v_9^\ell.$$

We next show properties of the trees T^ℓ and $\tilde{T}^\ell$ resulting form the iterative construction.

▶ **Lemma 17** (⋆). *The following are true for T^ℓ and $\tilde{T}^\ell$:*

- **(i)** *T^ℓ and $\tilde{T}^\ell$ are both non-crossing spanning trees,*
- **(ii)** *their near-near pairs are $(e_i^j, \tilde{e}_i^j)$, with associated gaps g_i^j, for $1 \le i \le 7$ and $1 \le j \le \ell$,*
- **(iii)** *for $1 \le j \le \ell$, the conflict graph has bidirected edges $g_7^j \leftrightarrow g_2^j \leftrightarrow g_1^j \leftrightarrow g_3^j \leftrightarrow g_5^j \leftrightarrow g_4^j \leftrightarrow g_6^j$,*
- **(iv)** *for $1 \le j < j' \le \ell$, the conflict graph has a bidirected edge $g_6^j \leftrightarrow g_7^{j'}$.*

▶ **Lemma 18.** *The conflict graph $H(T^\ell, \tilde{T}^\ell)$ has 7ℓ vertices and its largest acyclic sets have size at most $3\ell + 1$.*

Proof. That the conflict graph has 7ℓ vertices follows immediately from Lemma 17 (ii). Observe that from every bidirected path of length seven given in Lemma 17 (iii), we can pick at most four gaps for the acyclic set. Furthermore, the only way to pick four is to pick

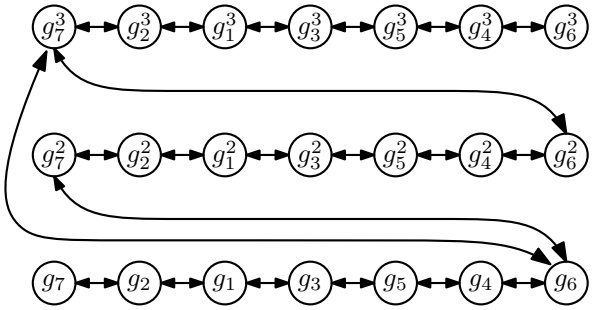


Figure 13 The relevant double conflicts in the conflict graph $H(T^\ell, \tilde{T}^\ell)$ for $\ell = 3$.

the gaps g_7^j, g_1^j, g_5^j and g_6^j, see also Figure 13. Now assume the cardinality of an acyclic set exceeds $3\ell + 1$. Then, by the pigeonhole principle, there exist at least two paths of length seven from which there are four pairs in the acyclic set. In particular, the set contains g_6^j and $g_7^{j'}$ for some $j < j'$ which are in double conflict by Lemma 17 (iv), contradicting acyclicity. ◀

Together, Lemma 18 and Theorem 5 imply the following.

▶ **Theorem 19.** *As a function of* n, $\operatorname{diam}(\mathcal{F}_n) \geq {}^{11}/_{7} \cdot n - o(n)$.

Proof. By Lemma 18 and Theorem 5 (ii), for any fixed ℓ, we have

$$\operatorname{diam}(\mathcal{F}_n) \geq \left(2 - \frac{3\ell+1}{7\ell}\right) n - c = \left(\frac{11}{7} - \frac{1}{7\ell}\right) n - c$$

for some c depending on ℓ. As a consequence, for any $\epsilon > 0$, $\operatorname{diam}(\mathcal{F}_n) \geq \left(\frac{11}{7} - \epsilon\right) n$ for sufficiently large n: Choose ℓ such that $\frac{1}{7\ell} < \epsilon$ and n large enough that $(\epsilon - \frac{1}{7\ell})n \geq c$, and calculate

$$\operatorname{diam}(\mathcal{F}_n) \geq \left(\frac{11}{7} - \frac{1}{7\ell}\right) n - c = \left(\frac{11}{7} - \epsilon\right) n + \left(\epsilon - \frac{1}{7\ell}\right) n - c \geq \left(\frac{11}{7} - \epsilon\right) n.$$

It follows that $\operatorname{diam}(\mathcal{F}_n) \geq \frac{11}{7} n - o(n)$. ◀

6 Conclusion

For spanning trees on convex point sets we proved that finding shortest flip sequences is NP-hard and we improved the current best lower bound on the diameter of the flip graph.

Interesting future directions include determining the exact diameters for flip graphs on point sets in convex as well as general position. We are also hopeful that our techniques for NP-hardness are fruitful in other settings. Very exciting is a recently announced breakthrough by Dorfer [20] concerning the complexity of computing the flip distance of triangulations on convex point sets, which is inspired by our methods.

References

1 Oswin Aichholzer, Franz Aurenhammer, and Ferran Hurtado. Sequences of spanning trees and a fixed tree theorem. *Computational Geometry: Theory & Applications*, 21(1-2):3–20, 2002. doi:10.1016/S0925-7721(01)00042-6.

2 Oswin Aichholzer, Brad Ballinger, Therese Biedl, Mirela Damian, Erik D. Demaine, Adam Hesterberg, Matias Korman, Anna Lubiw, Jayson Lynch, Josef Tkadlec, and Yushi Uno. Reconfiguration of non-crossing spanning trees. *Journal of Computational Geometry*, 15(1):224–253, 2024. doi:10.20382/jocg.v15i1a9.

3 Oswin Aichholzer, Sofia Brenner, Joseph Dorfer, Hung P. Hoang, Daniel Perz, Christian Rieck, and Francesco Verciani. Flipping odd matchings in geometric and combinatorial settings. In *Symposium on Graph Drawing and Network Visualization (GD)*, pages 12:1–12:18, 2025. doi:10.4230/LIPIcs.GD.2025.12.

4 Oswin Aichholzer and Joseph Dorfer. A linear time algorithm for the flip distance between plane spanning paths in convex point sets. *Computing in Geometry and Topology*, 5(3):2–1, 2026.

5 Oswin Aichholzer, Joseph Dorfer, and Birgit Vogtenhuber. Constrained flips in plane spanning trees. In *Symposium on Graph Drawing and Network Visualization (GD)*, pages 5:1–5:18, 2025. doi:10.4230/LIPIcs.GD.2025.5.

6 Oswin Aichholzer, Kristin Knorr, Wolfgang Mulzer, Johannes Obenaus, Rosna Paul, and Birgit Vogtenhuber. Flipping plane spanning paths. In *International Conference and Workshops on Algorithms and Computation (WALCOM)*, pages 49–60, 2023. doi:10.1007/978-3-031-27051-2_5.

7 Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discret. Comput. Geom.*, 54(2):368–389, 2015. doi:10.1007/S00454-015-9709-7.

8 Oswin Aichholzer and Klaus Reinhardt. A quadratic distance bound on sliding between crossing-free spanning trees. *Computational geometry*, 37(3):155–161, 2007. doi:10.1016/J.COMGEO.2004.12.010.

9 Selim G. Akl, Md Kamrul Islam, and Henk Meijer. On planar path transformation. *Information Processing Letters*, 104(2):59–64, 2007. doi:10.1016/j.ipl.2007.05.009.

10 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996. doi:10.1016/0166-218X(95)00026-N.

11 Carla Binucci, Fabrizio Montecchiani, Daniel Perz, and Alessandra Tappini. Flipping matchings is hard, 2025. doi:10.48550/arXiv.2503.02842.

12 Håvard Bakke Bjerkevik, Joseph Dorfer, Linda Kleist, Torsten Ueckerdt, and Birgit Vogtenhuber. Flip distance of non-crossing spanning trees: NP-hardness and improved bounds, 2026. arXiv:2603.22262.

13 Håvard Bakke Bjerkevik, Linda Kleist, Torsten Ueckerdt, and Birgit Vogtenhuber. Flipping non-crossing spanning trees. *Symposium on Discrete Algorithms (SODA)*, pages 2313–2325, 2025. doi:10.1137/1.9781611978322.77.

14 Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Computational Geometry: Theory & Applications*, 42(1):60–80, 2009. doi:10.1016/j.comgeo.2008.04.001.

15 Nicolas Bousquet, Lucas de Meyer, Théo Pierron, and Alexandra Wesolek. Reconfiguration of plane trees in convex geometric graphs. In *Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *LIPIcs*, pages 22:1–22:17, 2024. doi:10.4230/LIPIcs.SoCG.2024.22.

16 Nicolas Bousquet, Valentin Gledel, Jonathan Narboni, and Théo Pierron. A note on the flip distance between non-crossing spanning trees. *Computing in Geometry and Topology*, 2(1):8:1–8:7, 2023. doi:10.57717/cgt.v2i1.36.

17 Kevin Buchin, Valentin Polishchuk, Leonid Sedov, and Roman Voronov. Geometric secluded paths and planar satisfiability. In *Symposium on Computational Geometry (SoCG)*, pages 24–1, 2020. doi:10.4230/LIPIcs.SoCG.2020.24.

18 Jou-Ming Chang and Ro-Yu Wu. On the diameter of geometric path graphs of points in convex position. *Information Processing Letters*, 109(8):409–413, 2009. doi:10.1016/j.ipl.2008.12.017.

19 Satyan L Devadoss. A realization of graph associahedra. *Discrete Mathematics*, 309(1):271–276, 2009. doi:10.1016/J.DISC.2007.12.092.

20 Joseph Dorfer. Flip distance of triangulations of convex polygons/rotation distance of binary trees is NP-complete, 2026. arXiv:2602.22874.

21 David Eppstein. Happy endings for flip graphs. In *Symposium on Computational Geometry (SoCG 2007)*, pages 92–101, 2007. doi:10.1145/1247069.1247084.

22 David Eppstein. Happy endings for flip graphs. *Journal of Computational Geometry*, 1(1):3–28, 2010. doi:10.20382/JOCG.V1I1A2.

23 Stefan Felsner, Linda Kleist, Torsten Mütze, and Leon Sering. Rainbow cycles in flip graphs. In *Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *LIPIcs*, pages 38:1–38:14. Schloss Dagstuhl, 2018. doi:10.4230/LIPIcs.SOCG.2018.38.

24 Stefan Felsner, Linda Kleist, Torsten Mütze, and Leon Sering. Rainbow cycles in flip graphs. *SIAM Journal on Discrete Mathematics*, 34(1):1–39, 2020. doi:10.1137/18M1216456.

25 Carmen Hernando, Ferran Hurtado, Alberto Márquez, Mercè Mora, and Marc Noy. Geometric tree graphs of points in convex position. *Discrete Applied Mathematics*, 93(1):51–66, 1999. doi:10.1016/S0166-218X(99)00006-2.

26 M. Carmen Hernando, Ferran Hurtado, and Marc Noy. Graphs of non-crossing perfect matchings. *Graphs and Combinatorics*, 18(3):517–532, 2002. doi:10.1007/S003730200038.

27 Michael E. Houle, Ferran Hurtado, Marc Noy, and Eduardo Rivera-Campo. Graphs of triangulations and perfect matchings. *Graphs and Combinatorics*, 21(3):325–331, 2005. doi:10.1007/S00373-005-0615-2.

28 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999. doi:10.1007/PL00009464.

29 Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, Shun-ichi Maezawa, Yuta Nozaki, and Yoshio Okamoto. Hardness of finding combinatorial shortest paths on graph associahedra. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 261 of *LIPIcs*, pages 82:1–82:17, 2023. doi:10.4230/LIPIcs.ICALP.2023.82.

30 Linda Kleist, Peter Kramer, and Christian Rieck. On the connectivity of the flip graph of plane spanning paths. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 327–342, 2024. doi:10.1007/978-3-031-75409-8_23.

31 Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365–372, 1972. doi:10.1016/0012-365X(72)90093-3.

32 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry: Theory & Applications*, 49:17–23, 2015. doi:10.1016/J.COMGEO.2014.11.001.

33 Torrie L Nichols, Alexander Pilz, Csaba D Tóth, and Ahad N Zehmakan. Transition operations over plane trees. *Discrete Mathematics*, 343(8):111929, 2020. doi:10.1016/j.disc.2020.111929.

34 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. doi:10.1016/j.comgeo.2014.01.001.

35 Lionel Pournin. The diameter of associahedra. *Advances in Mathematics*, 259:13–42, 2014. doi:10.1016/j.aim.2014.02.035.

36 Daniel Dominic Sleator, Robert Endre Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. In *Symposium on Theory of Computing (STOC)*, pages 122–135, 1986. doi:10.1145/12130.12143.

37 Uli Wagner and Emo Welzl. Connectivity of triangulation flip graphs in the plane (part I: edge flips). In *Symposium on Discrete Algorithms (SODA)*, pages 2823–2841. SIAM, 2020. doi:10.1137/1.9781611975994.172.

38 Uli Wagner and Emo Welzl. Connectivity of triangulation flip graphs in the plane (part II: bistellar flips). In *Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *LIPIcs*, pages 67:1–67:16. Schloss Dagstuhl, 2020. doi:10.4230/LIPIcs.SOCG.2020.67.

39 Uli Wagner and Emo Welzl. Connectivity of triangulation flip graphs in the plane. *Discrete & Computational Geometry*, 68(4):1227–1284, 2022. doi:10.1007/S00454-022-00436-2.

Fréchet Distance in the Imbalanced Case

Lotte Blank
University of Bonn, Germany

Abstract

Given two polygonal curves P and Q defined by n and m vertices with $m \leq n$, we show that the discrete Fréchet distance in 1D cannot be approximated within a factor of $2-\varepsilon$ in $\mathcal{O}((nm)^{1-\delta})$ time for any $\varepsilon, \delta > 0$ unless OVH fails. Using a similar construction, we extend this bound for curves in 2D under the continuous or discrete Fréchet distance and increase the approximation factor to $1+\sqrt{2}-\varepsilon$ (resp. $3-\varepsilon$) if the curves lie in the Euclidean space (resp. in the L_∞-space). This strengthens the lower bound by Buchin, Ophelders, and Speckmann to the case where $m = n^\alpha$ for $\alpha \in (0,1)$ and increases the approximation factor of 1.001 by Bringmann. For the discrete Fréchet distance in 1D, we provide an approximation algorithm with optimal approximation factor and almost optimal running time. Further, for curves in any dimension embedded in any L_p space, we present a $(3+\varepsilon)$-approximation algorithm for the continuous and discrete Fréchet distance using $\mathcal{O}((n+m^2)\log n)$ time, which almost matches the approximation factor of the lower bound for the L_∞ metric.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Fréchet distance, SETH, Orthogonal Vectors, Lower Bounds, distance oracle, data structures

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.17

Related Version *Full Version*: https://arxiv.org/abs/2602.09551

Funding *Lotte Blank*: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 459420781 (FOR AlgoForGe).

1 Introduction

The Fréchet distance is a distance metric for polygonal curves, where each curve is represented as a piecewise-linear interpolation between its vertices. There are different variants of this metric and the two most common variants are the discrete and the continuous Fréchet distance. Alt and Godau [3] were the first to study the computational complexity of the Fréchet distance. Let n and m denote the number of vertices of the two curves, which we refer to as their complexities, with $n \geq m$. They showed that the continuous Fréchet distance can be computed in $\mathcal{O}(nm \log n)$ time. Eiter and Mannilla [21] defined the discrete Fréchet distance and gave an $\mathcal{O}(nm)$ time algorithm. Although both results were slightly improved later [15, 11, 1], Bringmann [6] showed that for d-dimensional curves with $d \geq 2$ it is very unlikely that significantly faster algorithms exists: unless the Strong Exponential Time Hypothesis (SETH) fails, no (1.001)-approximation algorithm for the continuous or discrete Fréchet distance can run in $\mathcal{O}((nm)^{1-\delta})$ time for any $\delta > 0$. Later, Bringmann and Mulzer [10] extended this result for the discrete Fréchet distance in 1D and increased the approximation factor up to $1.4-\varepsilon$ for any $\varepsilon > 0$[1]. For the continuous and the discrete Fréchet distance in 1D, Buchin, Ophelders, and Speckmann [12] showed a hardness result for approximation factors up to $3-\varepsilon$, but only in the case where $n = m$. A recent result by Blank and Driemel [5] shows that the continuous Fréchet distance in 1D can indeed be computed faster when $m = n^\alpha$ with $\alpha \in (0,1)$, namely in $\mathcal{O}(n \log n + m^2 \log^2 n)$ time.

[1] In [10], this result is stated only for $n = m$, but the same construction works for the case $n \neq m$ as well.

© Lotte Blank;
licensed under Creative Commons License CC-BY 4.0

Related Work. In recent years, many works have studied approximation algorithms for the Fréchet distance [18, 26, 27]. This line of research culminated in the first constant-factor approximation algorithm for the Fréchet distance that runs in subquadratic time, due to Cheng, Huang, and Zhang [17]. Their randomized $(7+\varepsilon)$-approximation algorithm for the continuous Fréchet distance runs in $\mathcal{O}(nm^{0.99}\log(n/\varepsilon))$ time for any $\varepsilon > 0$ and succeeds with probability at least $1 - 1/n^6$.

A simple idea to 3-approximate the Fréchet distance between curves where $m = n^\alpha$ with $\alpha \in (0,1)$ in subquadratic time is the following. Compute a curve P' of complexity $n' \in o(n)$ and its distance δ to P such that every curve Q of complexity m has Fréchet distance at least δ to P. Then compute the Fréchet distance δ' between P' and Q. Using triangle inequality, one can show that the Fréchet distance between P and Q is at least $\max\{\delta'/2, \delta\}$ and at most $3 \cdot \max\{\delta'/2, \delta\}$, which yields a 3-approximation. The difficulty of this approach is to compute such a curve P'. Two related variants of this simplification problem have been studied. The first is computing an optimal *m-simplification* of P, that is, a curve P^* of complexity at most m minimizing the Fréchet distance to P among all curves of complexity at most m. The second is computing an optimal *δ-simplification*, that is, a curve P^* of minimum complexity among all curves that have Fréchet distance at most δ to P. However, no previous result achieved a subquadratic running time for curves in arbitrary dimension when the approximation factor on the complexity of the simplified curve is constant [2, 4, 7, 13, 16, 22, 28].

A different approach to reduce the running time of computing the Fréchet distance is via Fréchet distance oracles. Such an oracle is a data structure that preprocesses an input curve and subsequently answers Fréchet distance queries with respect to query curves. Driemel and Har-Peled [19] described a data structure of size $\mathcal{O}(n\log n)$ using $\mathcal{O}(m^2\log n\log(m\log n))$ query time, which approximates the continuous Fréchet distance up to a (large) constant factor. For exact queries, [14] presented a data structure of size $\mathcal{O}(nm)^{\mathrm{poly}(d,m)}$ and query time in $\mathcal{O}((md)^{\mathcal{O}(1)}\log(nm))$, where the query complexity m is given at preprocessing time. Blank and Driemel [5] showed that for curves in 1D, there is a data structure of size $\mathcal{O}(n\log n)$ that computes the continuous Fréchet distance exactly in $\mathcal{O}(m^2\log^2 n)$ time. For the discrete Fréchet distance, Filtser and Filtser [22] improved [20] and presented a data structure with size in $\mathcal{O}(1/\varepsilon)^{kd}\log(\varepsilon)^{-1}$ that approximates the discrete Fréchet distance within a factor of $(1+\varepsilon)$ using expected $\widetilde{\mathcal{O}}(kd)$ query time, where $\widetilde{\mathcal{O}}(\cdot)$ hides polylogarithmic factors in n. Gudmundsson, Seybold, and Wong [24] used the lower bound construction by Bringmann [6] to show in a similar fashion to [8, 20, 25] the following. In 2D, there exists no Fréchet distance oracle with $\mathrm{poly}(n)$ preprocessing time and $\mathcal{O}((nm)^{1-\delta})$ query time and approximation factor 1.001 unless SETH fails.

Our Results. Our first set of results shows lower bounds for algorithms that approximate the Fréchet distance conditioned on the Orthogonal Vectors Hypothesis (OVH). We show that the discrete Fréchet distance between 1-dimensional curves cannot be $(2-\varepsilon)$-approximated in $\mathcal{O}((nm)^{1-\delta})$ time for any $\varepsilon, \delta > 0$ unless OVH fails. This increases the approximation factor of $1.4-\varepsilon$ by Bringmann and Mulzer [10]. Using a similar construction in 2D, we show that the continuous and discrete Fréchet distance cannot be approximated within a factor less than $1+\sqrt{2}$ (resp. 3) when the curves lie in the Euclidean space (resp. in the L_∞-space) in $\mathcal{O}((nm)^{1-\delta})$. This leads to lower bounds for Fréchet distance oracles as well and answers an open question by [24].

To obtain this first set of results, we reduce from the *Orthogonal Vector Problem:* Given two sets of vectors $U, V \subset \{0,1\}^d$ of size m and n, do there exist two vectors $u \in U$ and $v \in V$ such that $\langle u, v\rangle = 0$? Then, the results follow by OVH, which is implied by SETH [29, 9].

Table 1 Upper and Lower Bounds for the Fréchet distance Computation.

Upper Bounds						
	Discrete			Continuous		
1D	$\mathcal{O}(mn)$	exact	[21, 1]	$\widetilde{\mathcal{O}}(mn)$	exact	[3, 11, 15]
	$\widetilde{\mathcal{O}}(n+m^2)$	2-approx.	Cor.20	$\widetilde{\mathcal{O}}(n+m^2)$	exact	[5]
dD	$\mathcal{O}(mn)$	exact	[21, 1]	$\widetilde{\mathcal{O}}(mn)$	exact	[3, 11, 15]
	$\widetilde{\mathcal{O}}(n+m^2)$	$(3+\varepsilon)$-approx.	Thm.23	$\widetilde{\mathcal{O}}(n+m^2)$	$(3+\varepsilon)$-approx.	Thm.23
	$\widetilde{\mathcal{O}}(nm^{0.99})$	$(7+\varepsilon)$-approx.	[17]	$\widetilde{\mathcal{O}}(nm^{0.99})$	$(7+\varepsilon)$-approx.	[17]

Lower Bounds						
	Discrete			Continuous		
1D	$\nexists\ \mathcal{O}(m^{2-\delta})$	$(3-\varepsilon)$-approx.	[12]	$\nexists\ \mathcal{O}(m^{2-\delta})$	$(3-\varepsilon)$-approx.	[12]
	$\nexists\ \widetilde{\mathcal{O}}((mn)^{1-\delta})$	$(1.4-\varepsilon)$-approx.	[10]			
	$\nexists\ \widetilde{\mathcal{O}}((mn)^{1-\delta})$	$(2-\varepsilon)$-approx.	Thm.9			
2D		$\nexists\ \mathcal{O}((mn)^{1-\delta})$	(1.001)-approx.		[6]	
		$\nexists\ \mathcal{O}((mn)^{1-\delta})$	$(1+\sqrt{2}-\varepsilon)$-approx. in L_2		Thm.10	
		$\nexists\ \mathcal{O}((mn)^{1-\delta})$	$(3-\varepsilon)$-approx. in L_∞		Thm.10	

▶ **Definition 1** (Orthogonal Vectors Hypothesis (OVH)). *For all $\delta > 0$ there exists a $c > 0$ such that there is no algorithm solving OV instances $U, V \subset \{0,1\}^d$ of size m and n with $m \leq n$ and $d = c \log n$ in time $\mathcal{O}((nm)^{1-\delta})$.*

We construct a curve P of complexity $\mathcal{O}(n+m)$ depending on V and the size of U and a curve Q of complexity $\mathcal{O}(m)$ depending only on U. Then, we show that the Fréchet distance between P and Q is at most 1 if and only if the OV instance has a solution. In order to get a lower bound also for the case where $n \gg m$, we construct P such that the main part of P lies in a ball of radius 1. Hence, this main part can be matched to a single point of Q. Further, to obtain an inapproximability result of $2-\varepsilon$ in 1D, the points of P and Q have integer coordinates. Hence, there are only three different values that the vertices of the main part of P can have. This significantly restricts our possibilities to define P and is the main difference to the construction by Bringmann and Mulzer [10]. They also note in their paper that obtaining a lower bound for an approximation factor of 1.4 or higher requires a significantly different construction compared to theirs.

On the positive side, we present a data structure that preprocesses a 1-dimensional curve of complexity n in $\mathcal{O}(n \log n)$ time using $\mathcal{O}(m)$ space that 2-approximates the discrete Fréchet distance to query curves of complexity m in $\mathcal{O}(m^2 \log m)$ time. Hence, there exists a 2-approximation algorithm for the discrete Fréchet distance in $\mathcal{O}(n \log n + m^2 \log m)$ time matching our approximation factor lower bound and essentially the time lower bound by [12]. Secondly, we match our approximation factor lower bound in the L_∞-space by giving a simple $(3+\varepsilon)$-approximation algorithm for the discrete and continuous Fréchet distance using $\mathcal{O}((n+m^2)\log n)$ time for curves in any dimension under any L_p-norm. Table 1 summarizes the results of this paper and related work.

2 Preliminaries

For any two points $p_1, p_2 \in \mathbb{R}^d$, $\overline{p_1 p_2}$ denotes the directed line segment connecting p_1 with p_2. A polygonal curve P of complexity n is formed by ordered line segments $\overline{P(i)P(i+1)}$ of points $P(1), P(2), \ldots, P(n)$, where $P(i) \in \mathbb{R}^d$. It can be viewed as a function $P : [1, n] \to \mathbb{R}^d$,

where $P(i+\alpha) = (1-\alpha)P(i) + \alpha P(i+1)$ for $i \in \{1, \dots, n\}$ and $\alpha \in [0,1]$. This curve is also denoted with $\langle P(1), \dots, P(n)\rangle$. We call the points $P(i)$ vertices and the line segments $\overline{P(i)P(i+1)}$ edges of P. Further, $P[s,t]$ is the subcurve of P obtained from restricting the domain to the interval $[s,t]$. Two curves $P = \langle P(1), \dots, P(n)\rangle$ and $Q = \langle Q(1), \dots, Q(m)\rangle$ can be composed into the curve $P \circ Q = \langle P(1), \dots, P(n), Q(1), \dots, Q(m)\rangle$. For a number $k \in \mathbb{N}_0$, define P^k be the composition of k copies of P.

Let $P : [1,n] \to \mathbb{R}^d$ and $Q : [1,m] \to \mathbb{R}^d$ be two polygonal curves. A *continuous matching* M between P and Q is defined by two functions $h_P \in \mathcal{F}_n$ and $h_Q \in \mathcal{F}_m$ and consists of the tuples $(h_P(a), h_Q(a))$ for $a \in [0,1]$, where $\mathcal{F}_n$ is the set of all continuous, non-decreasing functions $h : [0,1] \to [1,n]$ with $h(0) = 1$ and $h(1) = n$, respectively $\mathcal{F}_m$ for m. Then, the *(continuous) Fréchet distance* between P and Q is defined as

$$d_{\mathrm{F}}(P,Q) = \min_{\text{continuous matching } M} \max_{(x,y)\in M} \|P(x) - Q(y)\|.$$

For a fixed δ, the *free space* $F_\delta(P,Q)$ is a subset of $[1,n] \times [1,m]$ such that $(x,y) \in F_\delta(P,Q)$ if and only if $\|P(x) - Q(y)\| \le \delta$. Points in $([1,n] \times [1,m]) \setminus F_\delta(P,Q)$ are contained in the non-free space. Hence, the continuous Fréchet distance between P and Q is at most δ if a continuous matching $M \subset F_\delta(P,Q)$ between P and Q exists.

A *discrete matching* M between P and Q is an ordered sequence that starts in $(1,1)$, ends in (n,m) and for any consecutive tuples (i,j), (i',j') in M it holds that $(i',j') \in \{(i+1,j), (i,j+1), (i+1,j+1)\}$. Then, the *discrete Fréchet distance* between P and Q is

$$d_{\mathrm{dF}}(P,Q) = \min_{\text{discrete matching } M} \max_{(i,j)\in M} \|P(i) - Q(j)\|.$$

The *free space matrix* M_δ is an $n \times m$ matrix, where $M_\delta[i,j] = 1$ if $|P(i) - Q(j)| \le \delta$ and otherwise $M_\delta[i,j] = 0$. We say that (i,j) is *reachable* if $d_{\mathrm{dF}}(P[1,i], Q[1,j]) \le \delta$.

▶ **Observation 2.** *Let P_1, P_2, Q_1, Q_2 be curves. If $d_F(P_1,Q_1) \le \delta$ and $d_F(P_2,Q_2) \le \delta$, then $d_F(P_1 \circ P_2, Q_1 \circ Q_2) \le \delta$. For the discrete Fréchet distance, the analogue statement holds.*

▶ **Observation 3.** *Let P and Q be polygonal curves, then $d_F(P,Q) \le d_{dF}(P,Q)$.*

3 Approximation lower bounds for the imbalanced case

In this section, we show a lower bound for the discrete Fréchet distance in 1D and for the discrete and continuous Fréchet distance in 2D depending on OVH. The construction for both bounds is very similar and we start with giving a schematic view of the reduction.

Let $U, V \subset \{0,1\}^d$ be two sets of size m and n, where $m \le n$ and denote with v_{ik} (resp. u_{ik}) the k-th bit of the i-th vector of V (resp. U). We construct two 1-dimensional (resp. 2-dimensional) curves P and Q of complexity $\mathcal{O}(nd)$ and $\mathcal{O}(md)$, where P depends on V and Q on U. We show that their discrete (and continuous) Fréchet distance is at most 1 if and only if two vectors that are orthogonal exist in V and U. Define

$$P = P_s \circ (P^*)^{m-1} \circ P_{null} \circ \bigcirc_{i=1}^{n} \left((P_0)^{d+1} \circ P_1 \circ \bigcirc_{\ell=1}^{d} P_{v_{i\ell}} \circ P_1\right) \circ P_{null} \circ (P^*)^{m-1} \circ P_s,$$

$$Q = (Q^*)^{m-1} \circ \bigcirc_{j=1}^{m} \left(Q_{null} \circ Q_c \circ (Q_1)^{d+1} \circ Q_0 \circ \bigcirc_{\ell=1}^{d} Q_{u_{j\ell}} \circ Q_0 \circ Q_c\right) \circ Q_{null} \circ (Q^*)^{m-1},$$

where the subcurves are defined later. Since the complexity of P can be arbitrarily larger than the complexity of Q, we construct the curve P such that the main part of P, i.e., in between the two subcurves P_{null}, is contained in a ball of radius 1. Then, we can match this whole part to a single point on Q. This point is Q_c in our construction. At a high

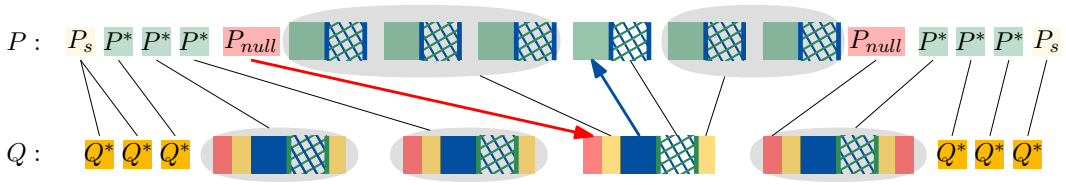


Figure 1 A schematic view of the construction of the curves P and Q. Here, Q_{null} is drawn in red, P_0 and Q_0 in green, P_1 and Q_1 in blue, and Q_c in orange.

level idea, we want to enforce that in a matching that realizes the Fréchet distance, the two subcurves P_{null} get roughly matched to different subcurves Q_{null}. Then, the subcurve in between the two points matched to P_{null} contains $(Q_1)^{d+1}$ as a subcurve. Now, we enforce that $(Q_1)^{d+1}$ gets roughly matched to $(P_0)^{d+1}$. Hence, afterwards $\bigcirc_{k=1}^{d} P_{v_{ik}}$ and $\bigcirc_{\ell=1}^{d} Q_{u_{j\ell}}$ have to be matched together for some i and j. We will show that this is only possible if $\langle v_i, u_j \rangle = 0$. This matching is schematically visualized in Figure 1. We define

$$P_s = \langle (-1,0) \rangle, \quad P_{null} = \langle (-5,0) \rangle, \quad P^* = \langle (-3,0) \rangle \circ \langle P_0 \rangle^{2d+3} \circ \langle (-3,0) \rangle,$$
$$Q_c = \langle (0,0) \rangle, \quad Q_{null} = \langle (-4,0) \rangle, \quad Q^* = \langle (-2,0), (0,0), (-2,0) \rangle,$$

for 2D and the definition carries over directly to 1D. The curves P_0, P_1, Q_0, and Q_1 are visualized in Figure 2. For the 1D setting, let

$$P_0 = \langle -1, 1 \rangle, \quad P_1 = \langle -1, 0 \rangle, \quad Q_0 = \langle -2, 1 \rangle, \quad Q_1 = \langle -2, 2 \rangle.$$

For the 2D constructions, define eight points with distance 1 to $(0,0)$ where $a_1 = (-1,0)$ and the direction of a_{i+1} is the direction of a_i rotated by $\pi/4$, i.e., in L_2

$$a_1 = (-1,0), \quad a_2 = (-1/\sqrt{2}, -1/\sqrt{2}), \quad a_3 = (0,-1), \quad a_4 = (1/\sqrt{2}, -1/\sqrt{2}),$$
$$a_5 = (1,0), \quad a_6 = (1/\sqrt{2}, 1/\sqrt{2}), \quad a_7 = (0,1), \quad a_8 = (-1/\sqrt{2}, 1/\sqrt{2}),$$

and in L_∞

$$a_1 = (-1,0), \quad a_2 = (-1,-1), \quad a_3 = (0,-1), \quad a_4 = (1,-1),$$
$$a_5 = (1,0), \quad a_6 = (1,1), \quad a_7 = (0,1), \quad a_8 = (-1,1).$$

Define $c = \sqrt{2}$ and $\tilde{a} = a_8 + a_7$ in L_2 and $c = 2$ and $\tilde{a} = 2a_8$ in L_∞. Note that c is defined such that ca_1 is the point with distance 1 to a_8 and a_2 that maximizes the distance to $(0,0)$ and $\tilde{a}$ such that $\|\tilde{a} - a_2\| = 1 + c$ and $\|\tilde{a} - a_8\| = 1$. Then we set

$$P_0 = \langle a_1, a_2, \dots a_7, a_8, a_7 \dots, a_1 \rangle, \quad P_1 = \langle a_1, a_2, \dots, a_8, a_1, a_2, a_1 a_8, \dots, a_1 \rangle,$$
$$Q_0 = \langle 2a_1, 2a_2, \dots, 2a_8, ca_1, 2a_8, \dots, 2a_1 \rangle, \quad Q_1 = \langle 2a_1, 2a_2, \dots, 2a_7, \tilde{a}, 2a_7, \dots, 2a_1 \rangle.$$

We show that this results in an inapproximability result of $(1 + c - \varepsilon)$ for time in $\mathcal{O}((nm)^{1-\delta})$.

▶ **Lemma 4.** *If there exist indices i and j such that $\langle v_i, u_j \rangle = 0$, then in every construction for P and Q above it holds that $d_{dF}(P, Q) \leq 1$.*

Proof. For every construction of P and Q, it holds that $d_{\mathrm{dF}}(P^*, Q^*) \leq 1$, $d_{\mathrm{dF}}(P_s, Q^*) \leq 1$. Further, for $x, y \in \{0, 1\}$ with $\langle x, y \rangle = 0$, it holds that $d_{\mathrm{dF}}(P_x, Q_y) \leq 1$. Then, we can match the following with Fréchet distance at most 1 and the lemma follows by Observation 2

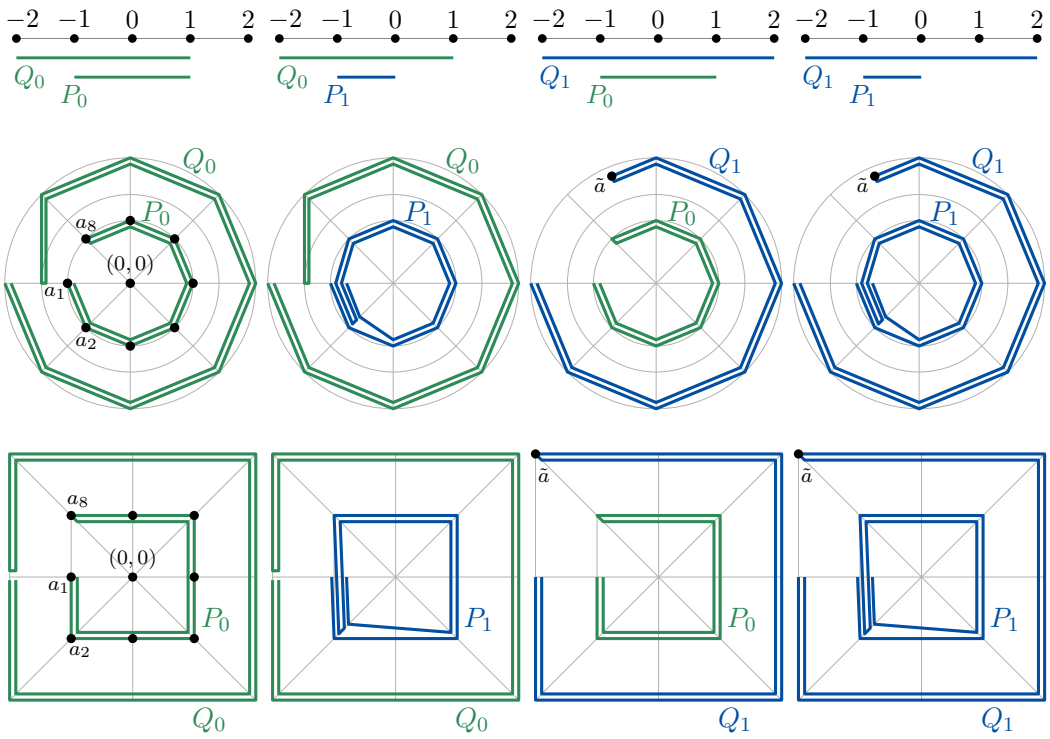


Figure 2 The curves P_0, P_1, Q_0, and Q_1. The upper curves are defined in 1D, the middle curves for the L_2-metric in 2D and the lower curves for the L_∞-metric in 2D.

$P_s \circ (P^*)^{m-j}$	and	$(Q^*)^{m-1}$,
$(P^*)^{j-1} \circ P_{null}$	and	$\bigcirc_{k=1}^{j-1} (Q_{null} \circ Q_c \circ (Q_1)^{d+1} \circ Q_0 \circ \bigcirc_{\ell=1}^{d} Q_{u_{k\ell}} \circ Q_0 \circ Q_c) \circ Q_{null}$,
$\bigcirc_{k=1}^{i-1} \left((P_0)^{d+1} \circ P_1 \circ \bigcirc_{\ell=1}^{d} P_{v_{k\ell}} \circ P_1\right)$	and	Q_c,
$(P_0)^{d+1} \circ P_1 \circ \bigcirc_{\ell=1}^{d} P_{v_{i\ell}} \circ P_1$	and	$(Q_1)^{d+1} \circ Q_0 \circ \bigcirc_{\ell=1}^{d} Q_{u_{j\ell}} \circ Q_0$,
$\bigcirc_{k=i+1}^{n} \left((P_0)^{d+1} \circ P_1 \circ \bigcirc_{\ell=1}^{d} P_{v_{k\ell}} \circ P_1\right)$	and	Q_c,
$P_{null} \circ (P^*)^{m-j}$	and	$\bigcirc_{k=j+1}^{m} (Q_{null} \circ Q_c \circ (Q_1)^{d+1} \circ Q_0 \circ \bigcirc_{\ell=1}^{d} Q_{u_{k\ell}} \circ Q_0 \circ Q_c) \circ Q_{null}$,
$(P^*)^{j-1} \circ P_s$	and	$(Q^*)^{m-1}$. ◀

To show the other direction, we first prove some lemmas.

▶ **Lemma 5.** *Consider the 1D setting and let $Q' = \bigcirc_{\ell=1}^{d'} Q_{\sigma_\ell}$, where $\sigma_\ell \in \{0,1\}$ for every ℓ. Further, let $s \leq t$ be such that $P[s,t]$ is a subcurve in between the two subcurves P_{null} and $d_{dF}(Q', P[s,t]) < 2$. Then $P[s,t] = \bigcirc_{\ell=1}^{d'} P_{\pi_\ell}$ where $\pi_\ell = 0$ if $\sigma_\ell = 1$ and $\pi_\ell \in \{0,1\}$ if $\sigma_\ell = 0$ for every ℓ.*

Proof. It holds that every second vertex of Q' is -2 and the vertices of value -1 on $P[s,t]$ are the only vertices on $P[s,t]$ within distance less than 2 to those vertices of Q'. Further, it holds that the vertices with value -1 of $P[s,t]$ have distance less than two only to the vertices of value -2 of Q'. Hence, since the vertices of value -1 on $P[s,t]$ are exactly every second vertex on $P[s,t]$, we must traverse $P[s,t]$ and Q' synchronously. Therefore, if $\sigma_\ell = 0$ then $\pi_\ell \in \{0,1\}$ and if $\sigma_\ell = 1$ then $\pi_\ell = 0$ by construction. ◀

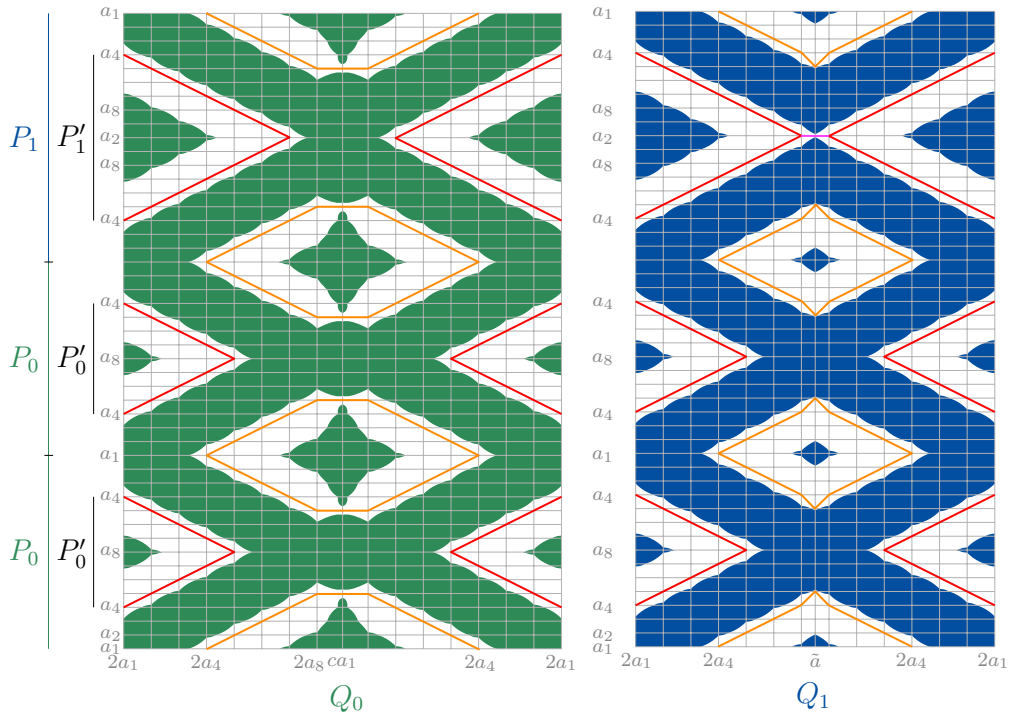


Figure 3 In white: the non-free space of the curve Q_0 and $P_0 \circ P_0 \circ P_1$ on the left and of the curve Q_1 and $P_0 \circ P_0 \circ P_1$ on the right. The red, orange and pink paths visualize the proof of Lemma 6.

In 2D, we also want to show a bound for the continuous Fréchet distance. Then the vertices of Q can also be matched to points on edges of P. We use the notation of the following subcurve of P_0 (resp. P_1): $P_0' = \langle a_4, a_5, \ldots, a_8, \ldots, a_4 \rangle$ (resp. $P_1' = \langle a_4, \ldots, a_8, a_1, a_2, a_1, a_8, \ldots, a_4 \rangle$).

▶ **Lemma 6.** *Consider the 2D setting. For any subcurve P' of P in between the two subcurves P_{null} with $d_F(P', Q_y) < 1 + c$ for $y \in \{0, 1\}$, it holds that P_x' is a subcurve of P' and P' is a subcurve of $\langle a_4, a_3, a_2 \rangle \circ P_x \circ \langle a_2, a_3, a_4 \rangle$, where $x = 0$ if $y = 1$ and $x \in \{0, 1\}$ if $y = 0$.*

Proof. We show this lemma by constructing paths in the non-free space for $\delta = 1 + c$ (see Figure 3). First observe that $\|2a_i - a_j\| \geq 1 + c$ for any $i, j \in \{1, \ldots, 8\}$ with $i - j \geq 3 \mod 8$.

Assume that the first (resp. last) vertex of Q_y is matched to a point on a subcurve P_x' with $x \in \{0, 1\}$. Then, there must exist a path completely contained in the free space ending at the last (resp. first) vertex of Q_y. This cannot be the case since there exists a continuous matching between P_x' and $\langle 2a_1, 2a_2 \ldots, 2a_5, \ldots, 2a_1 \rangle$ that is completely contained in the non-free space (see the red paths in Figure 3). Therefore, the first as well as the last vertex of Q_y has to be matched to a point on a subcurve P'' of P such that $P'' = \langle a_4, a_3, a_2, a_1, a_2, a_3, a_4 \rangle$.

Next, we show that P' contains exactly one P_x' as a subcurve. Since $\|ca_1 - a_5\| \geq 1 + c$ (resp. $\|\tilde{a} - a_5\| \geq 1 + c$), there exists a path that is completely contained in the non-free space between $\langle a_5, a_4, \ldots a_2, a_1, a_2, \ldots, a_4, a_5 \rangle$ and $\langle 2a_8, ca_1, 2a_8, 2a_7, \ldots, 2a_4, 2a_5, \ldots, 2a_8, ca_1, 2a_8 \rangle$ (resp. $\langle \tilde{a}, 2a_7, \ldots, 2a_4, 2a_5, \ldots, \tilde{a} \rangle$) (see the orange paths in Figure 3). Hence by construction of P, it follows that P_x' is a subcurve of P' and P' is a subcurve of $\langle a_4, a_3, a_2 \rangle \circ P_x \circ \langle a_2, a_3, a_4 \rangle$ with $x \in \{0, 1\}$. It remains to show that $x \neq 1$, if $y = 1$. This follows since the distance between a_2 and $2a_7$, $\tilde{a}$, $2a_7$ is at least $1 + c$ (see the pink path in Figure 3). ◀

The next lemma follows directly by the construction of the curves and Lemma 6.

▶ **Lemma 7.** *Consider the 2D setting and let* $Q' = \bigcirc_{\ell=1}^{d'} Q_{\sigma_\ell}$*, where* $\sigma_\ell \in \{0,1\}$ *for every* ℓ*. Further, let* $s \le t$ *be such that* $P[s,t]$ *is a subcurve in between the two subcurves* P_{null} *and* $d_F(Q', P[s,t]) < 1 + c$*. Then,* $P'_{\pi_1} \circ \bigcirc_{\ell=2}^{d'-1} P_{\pi_\ell} \circ P'_{\pi_{d'}}$ *is a subcurve of* $P[s,t]$ *and* $P[s,t]$ *is a subcurve of* $\langle a_4, a_3, a_2\rangle \circ \bigcirc_{\ell=1}^{d'} P_{\pi_\ell} \circ \langle a_2, a_3, a_4\rangle$ *where* $\pi_\ell = 0$ *if* $\sigma_\ell = 1$ *and* $\pi_\ell \in \{0,1\}$ *if* $\sigma_\ell = 0$ *for every* ℓ*.*

▶ **Lemma 8.** *If* $d_{dF}(P,Q) < 2$ *(resp.* $d_F(P,Q) < 1 + c$*) in the 1D (resp. 2D) construction, then there exist two indices* i *and* j *such that* $\langle v_i, u_j\rangle = 0$*.*

Proof. Let M be a discrete (resp. continuous) matching that realizes the discrete (resp. continuous) Fréchet distance between P and Q. Assume that $d_{\mathrm{F}}(P,Q) < 2$ (resp. $d_{\mathrm{dF}}(P,Q) < 1 + c$), then in the matching M, the two points of value P_{null} of P must be matched to two distinct points q_1, q_2 on Q both on an edge that contains the value Q_{null}, since they are the only points within distance at most $1 + c \le 3$ to P_{null}. Let $P'_0 = P_0$ and $P'_1 = P_1$ in the 1D setting. Define j such that q_1 is contained on an incident edge to the j-th Q_{null} on Q. Then, the first $(Q_1)^{d+1}$ after the j-th Q_{null} on Q must be matched to a subcurve P' of P such that $P'_1 \circ (P_0)^{d+1} \circ P'_1$ is a subcurve of P' by Lemma 5 and Lemma 7. The only time we have $(P_0)^{d+1}$ in P is before $P_1 \circ \bigcirc_{\ell=1}^{d} P_{v_{i\ell}}$ for an i. Hence by Lemma 5 and Lemma 7, there exists an index i such that in the matching M the subcurve $P'_{v_{i,1}} \circ \bigcirc_{\ell=2}^{d-1} P_{v_{i,\ell}} \circ P'_{v_{i,d}}$ is matched to a subcurve of $\bigcirc_{\ell=1}^{d} Q_{u_{j,\ell}}$ and $v_{i,\ell} = 0$ if $u_{j,\ell} = 1$. Hence, it holds that $\langle v_i, u_j\rangle = 0$. ◀

By Lemma 4 and Lemma 8 together with Observation 3, the next two theorems follow.

▶ **Theorem 9.** *For any* $\delta, \varepsilon > 0$*, there does not exist a* $(2-\varepsilon)$*-approximation algorithm for the discrete Fréchet distance between 1-dimensional curves of complexity* n *and* m *using* $\mathcal{O}((nm)^{1-\delta})$ *time unless OVH fails.*

▶ **Theorem 10.** *For any* $\delta, \varepsilon > 0$*, there does not exist a* $(k-\varepsilon)$*-approximation algorithm for the continuous or discrete Fréchet distance between 2-dimensional curves of complexity* n *and* m *using* $\mathcal{O}((nm)^{1-\delta})$ *time unless OVH fails, where* $k = 3$ *(resp.* $k = 1 + \sqrt{2} \approx 2.41$*) if the curves lie in the* L_∞*-space (resp. in the Euclidean space).*

Our construction gives also a bound on Fréchet distance oracles (proof in the full version).

▶ **Theorem 11.** *Assume OVH holds true. For any* $\delta > 0$*, there exists a* $c > 0$ *such that for any* n *and* $m > c \log n$ *there is no data structure storing a curve of complexity* n *with* $\mathrm{poly}(n)$ *preprocessing time, that when given a query curve of at most complexity* m *can answer continuous or discrete Fréchet distance queries in* $\mathcal{O}((nm)^{1-\delta})$ *query time within the same approximation factor as in Theorem 9 and Theorem 10.*

4 2-Approximation for the discrete Fréchet distance in 1D

Given a value m and a 1-dimensional curve P of complexity $n \ge m$. We want to preprocess P in near-linear time and store information about P using $\mathcal{O}(m)$ space such that for any 1-dimensional query curve Q of complexity at most m we can approximate the discrete Fréchet distance between P and Q in $\mathcal{O}(m^2 \log m)$ time.

Preprocessing. Define $\delta_m = \min\{d_{\mathrm{dF}}(P,Q) \mid Q \text{ has complexity } m\}$. We want to compute a curve P' such that $d_{\mathrm{dF}}(P,P') \le \delta_m/2$. The complexity of P' might be n, but it should be possible to store a compact description of P' in $\mathcal{O}(m)$ space. To compute P', we first compute the optimal m-simplification P^* of P for the discrete Fréchet distance. The curve P^* is

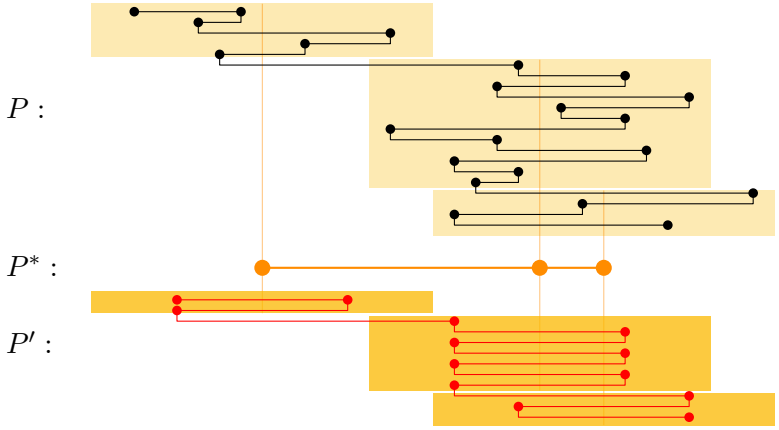


Figure 4 The black curve is P and the middle curve is an optimal 3-simplification for P. The red curve below is the compressed curve P'. Here, $k_1 = 3$, $k_2 = 7$, $k_3 = 3$ and $z_1 = -1$, $z_2 = -1$, $z_3 = 1$.

computed via searching over the possible values δ for δ_m and then computing the optimal δ-simplification. We compute the optimal δ-simplification of P using Algorithm 1. It begins by finding the longest prefix $P[1, i_1]$ that is contained in a 2δ-interval, whose center becomes the first vertex of the δ-simplification. Then, it continues by finding the largest index i_2 such that $P[i_1 + 1, i_2]$ is contained in a 2δ-interval and again the center of this interval is added to the δ-simplification. This process is repeated until the final vertex of P is reached.

Algorithm 1 Optimal δ-simplification of P.

$\ell \leftarrow 1$, $a \leftarrow P(1)$, $b \leftarrow P(1)$
for $i = 1$ *to* n **do**
  if $P(i) \leq a + 2\delta$ *and* $b - 2\delta \leq P(i)$ **then**
    $a \leftarrow \min\{a, P(i)\}$, $b \leftarrow \max\{b, P(i)\}$
  else
    $p_\ell \leftarrow \frac{a+b}{2}$, $a \leftarrow P(i)$, $b \leftarrow P(i)$, $i_\ell \leftarrow i - 1$, $\ell \leftarrow \ell + 1$
Return $P_\delta = \langle p_1, \ldots, p_\ell \rangle$, $i_0 = 0, i_1, \ldots, i_\ell = n$

The proof of the following lemma is in the full version.

▶ **Lemma 12.** *Algorithm 1 computes an optimal δ-simplification P_δ of P using $\mathcal{O}(n)$ time.*

If the complexity of the δ-simplification is greater than m, it holds that $\delta_m > \delta$. Otherwise, $\delta_m \leq \delta$. In contrast to higher dimensions, the optimal m-simplification for any m of a 1-dimensional curve P has distance $\delta_m \in \{|P(i) - P(j)|/2 \mid i, j \in \{1, \ldots, n\}\}$ to P, where the distance is only defined by two vertices of P. By Frederickson and Johnson [23], we can search over all possible values for δ_m using $\mathcal{O}(\log n)$ calls of Algorithm 1 and get the following lemma.

▶ **Lemma 13.** *We can compute an optimal m-simplification P^* of P together with a discrete matching that realizes the discrete Fréchet distance between P^* and P in $\mathcal{O}(n \log n)$ time.*

Let $P^* = \langle p_1, \ldots, p_\ell \rangle$ be an optimal m-simplification of P. Let M be the discrete matching between P and P^* computed by Algorithm 1, and $P(i_j)$ be the last vertex of P that is matched to p_j; i.e. $M = ((1,1),(2,1)\ldots,(i_1,1),(i_1+1,2),(i_1+2,2),\ldots,(i_2,2),(i_2+1,3),\ldots,(i_\ell,\ell))$.

We construct a curve P' that has discrete Fréchet distance at most $\delta_m/2$ to P and can be stored in a compressed way using $\mathcal{O}(m)$ space (see Figure 4). We call P' a *compressed simplification*. For every $(x, y) \in M$, define $r_x = p_y - \delta_m/2$ if $P(x) < p_y$ and $r_x = p_y + \delta_m/2$ otherwise. Then, it holds that $|r_x - P(x)| \leq \delta_m/2$. The compressed simplification P' is $\langle r_1, \ldots, r_n \rangle$ after deleting the vertices with identical predecessor, so that P' contains r_i only if $r_i \neq r_{i-1}$. For every j, we count the total number k_j that P' contains $p_j \pm \delta_m/2$ and define $z_j \in \{-1, 1\}$ such that the first time P' contains $p_j + z_j\delta_m/2$ is before P' contains $p_j - z_j\delta_m/2$. Therefore, the values $p_1, \ldots, p_\ell$, $k_1, \ldots, k_\ell$, $z_1, \ldots, z_\ell$, and δ_m fully describe P'. The next lemma follows directly by construction and $\ell \leq m$.

▶ **Lemma 14.** *Given P and its optimal m-simplification P^*, we can compute in $\mathcal{O}(n)$ time a curve P' such that $d_{dF}(P, P') \leq \delta_m/2$ and P' can be stored using $\mathcal{O}(m)$ space.*

The Query Algorithm. In the following, we give a 2-approximation query algorithm for the discrete Fréchet distance between P and any query curve Q of complexity m. We use the precomputed compressed simplification P' and compute $d_{\mathrm{dF}}(P', Q)$ if it is at least $\frac{3}{2}\delta_m$.

▶ **Lemma 15.** *If $d_{dF}(P', Q) \leq \frac{3}{2}\delta_m$, then $d_{dF}(P, Q) \in [\delta_m, 2\delta_m]$. Otherwise, define δ such that $\frac{3}{2}\delta = d_{dF}(P', Q)$. Then, $d_{dF}(P, Q) \in [\delta, 2\delta)$.*

Proof. By definition of δ_m, it holds that $d_{\mathrm{dF}}(P, Q) \geq \delta_m$ for every curve Q of complexity m. Further, by Lemma 14 it holds that $d_{\mathrm{dF}}(P, P') \leq \delta_m/2$. Hence, if $d_{\mathrm{dF}}(P', Q) \leq \frac{3}{2}\delta_m$, then

$$\delta_m \leq d_{\mathrm{dF}}(P, Q) \leq d_{\mathrm{dF}}(P', Q) + d_{\mathrm{dF}}(P, P') \leq \frac{3}{2}\delta_m + \frac{1}{2}\delta_m \leq 2\delta_m.$$

Otherwise, it holds that $\frac{3}{2}\delta = d_{\mathrm{dF}}(P', Q) > \frac{3}{2}\delta_m$. Hence, $\delta > \delta_m$. Therefore,

$$\delta < \frac{3}{2}\delta - \frac{1}{2}\delta_m \leq d_{\mathrm{dF}}(P', Q) - d_{\mathrm{dF}}(P', P) \leq d_{\mathrm{dF}}(P, Q) \leq d_{\mathrm{dF}}(P', Q) + d_{\mathrm{dF}}(P', P) < 2\delta.$$ ◀

Using Lemma 15, we could simply compute $d_{\mathrm{dF}}(P', Q)$ to get a 2-approximation algorithm for $d_{\mathrm{dF}}(P, Q)$. However, computing $d_{\mathrm{dF}}(P', Q)$ with known discrete Fréchet distance algorithms takes roughly $\mathcal{O}(m \cdot \sum_{i=1}^{\ell} k_i)$ time and $\sum_{i=1}^{\ell} k_i$ could be n. In the following, we make use of the special structure of P' to get a faster algorithm that decides whether $d_{\mathrm{dF}}(P', Q) \leq \frac{3}{2}\delta$ for any value $\delta \geq \delta_m$. We use the concept of the free space matrix $M_{(3/2)\delta}$, but do not compute all reachable entries explicitly. Define $s_i = \sum_{j=1}^{i-1} k_j + 1$ to be the index of the first vertex with value $p_i \pm \delta_m/2$ in P' and $s_{\ell+1} = |P'| + 1$. Algorithm 2 iterates over the sub-curves $P'[s_i, s_{i+1} - 1]$ of P' from $i = 1$ to ℓ and stores in the set F_{pre} the indices j of the vertices of Q such that $(s_i - 1, j)$ is reachable (see the hatched cells in Figure 5). Using F_{pre}, in iteration i we successively compute the set F_{new} of indices such that $(s_{i+1} - 1, j)$ is reachable and make use of the following property.

▶ **Observation 16.** *For any point $q \in \mathbb{R}$, it holds that exactly one of the following is true:*

- *$|q - p_i| \leq \delta$ and $|q - P'(s_i + x)| \leq \frac{3}{2}\delta$ for all $x \in \{0, \ldots, k_i - 1\}$, or*
- *$|q - p_i| \in (\delta, 2\delta]$ and $|q - P'(s_i + x)| \leq \frac{3}{2}\delta$ for all even (resp. odd) $x \in \{0, \ldots, k_i - 1\}$ and $|q - P'(s_i + x)| > \frac{3}{2}\delta$ for all odd (resp. even) $x \in \{0, \ldots, k_i - 1\}$, or*
- *$|q - p_i| > 2\delta$.*

By Observation 16, it holds that whenever $|Q(j) - p_i| \leq \delta$, there exists an number a such that $(s_i + x, j)$ with $x \in \{0, \ldots, k_i - 1\}$ is reachable if and only if $x \geq a$ and whenever $|Q(j) - p_i| > 2\delta$, then $(s_i + x, j)$ is not reachable for any integer $x \in \{0, \ldots, k_i - 1\}$ and we

set $a = \infty$. When $|Q(j) - p_i| \in (\delta, 2\delta]$, things get slightly more complicated. However, we can still store a number a such that $(s_i + x, j)$ is reachable for all $x \in \{a, a+2, \dots\}$ with $x \leq k_i - 1$. Since there might be more reachable pairs $(s_i + x, j)$, we store a set A of size at most m and a number u which can be used to reconstruct all other reachable pairs $(s_i + x, j)$ with $x \in \{0, \dots, a\}$. Then, let $B \subset \{0, \dots, \min\{k_i - 1, a\}\}$ be the set such that for all $x \leq \min\{k_i - 1, a\}$ it holds that $(s_i + x, j - 1)$ is reachable if and only if $x \in B$. It holds that $B \neq \emptyset$ only if $|Q(j-1) - p_i| \in (\delta, 2\delta]$. In that case, define $z = 0$ (resp. $z = 1$) if $Q(j-1)$ and $Q(j)$ lie on the same side (resp. different sides) of p_i. Then for any $x + z > 0$, it holds that that $(s_i + x + z, j - 1)$ is reachable if and only if $x \in B$. Using this property, we store a counter u that counts the sum of the values z until we reach again a vertex with $|Q(j') - p_i| \notin (\delta, 2\delta]$. If $x + z = 0$, we additionally check whether $(s_i - 1, j - 1)$ or $(s_i - 1, j)$ is reachable by looking at the set F_{pre}. If one of them is reachable, then $(s_i, j - 1) = (s_i + (u - u), j)$ is reachable and we add the current number u to the stored set A. In short, we intend to maintain the following invariant. During iteration i of the outer for-loop and at the end of iteration j of the inner for-loop, it holds that $(s_i + x, j)$ is reachable with $x \in \{0, \dots, k_i - 1\}$ if and only if

a) $x \in \{u - z \mid z \in A\} \cup \{a\}$ or **b)** $x > a$ and $|Q(j) - P'(s_i + x)| \leq (2/3)\delta$.

See Figure 5 for an example. Here, during iteration $i = 3$ of the outer for-loop and at the end of iteration $j = 7$ (resp. $j = 8$) of the inner for-loop, it holds that $A = [0, 2]$, $u = 2$, and $a = \infty$ (resp. $A = [\,]$, $u = 0$, and $a = 0$).

Algorithm 2 Decision Algorithm for $d_{\text{dF}}(P', Q) \leq \frac{3}{2}\delta$.

```
1  F_new ← {0}
2  for i = 1 to ℓ do
3      A ← [ ], a ← ∞, u ← 0, F_pre ← F_new, F_new ← [ ]
4      for j = 1 to m do
5          if |Q(j) − p_i| > 2δ then
6              A ← [ ], a ← ∞, u ← 0
7          if |Q(j − 1) − Q(j)| > 2δ and |Q(j) − p_i|, |Q(j − 1) − p_i| ∈ (δ, 2δ] then
8              a ← a + 1, u ← u + 1
9          if (j ∈ F_pre or j − 1 ∈ F_pre) and |P'(s_i) − Q(j)| ≤ (3/2)δ then
10             Append u to A
11         if |Q(j) − p_i| ≤ δ then
12             a ← min{a, u − min{{x | x ∈ A}} ∪ {∞}}
13             A ← [ ], u ← 0
14         Delete all x ∈ A with x ≤ u − k_i
15         if |P'(s_{i+1} − 1) − Q(j)| ≤ (3/2)δ and (a < k_i or u − k_i + 1 ∈ A) then
16             F_new ← j
```

▶ **Lemma 17.** *Assume that at the beginning of iteration i of the outer for-loop after setting $F_{pre} \leftarrow F_{new}$, it holds $(s_i - 1, j)$ is reachable if and only if $j \in F_{pre}$. Then, at the end of iteration i it holds that $(s_{i+1} - 1, j)$ is reachable if and only if $j \in F_{new}$.*

Proof. Denote with A_j, a_j, and u_j the value of A, a, and u at the end of iteration j in the inner for-loop of Algorithm 2. Then, we show that the following invariant is true; for every $x \in \{0, \dots, k_i - 1\}$, it holds that $(s_i + x, j)$ is reachable if and only if

a) $x \in \{u_j - z \mid z \in A_j\} \cup \{a_j\}$ or **b)** $x > a_j$ and $|Q(j) - P'(s_i + x)| \leq (2/3)\delta$.

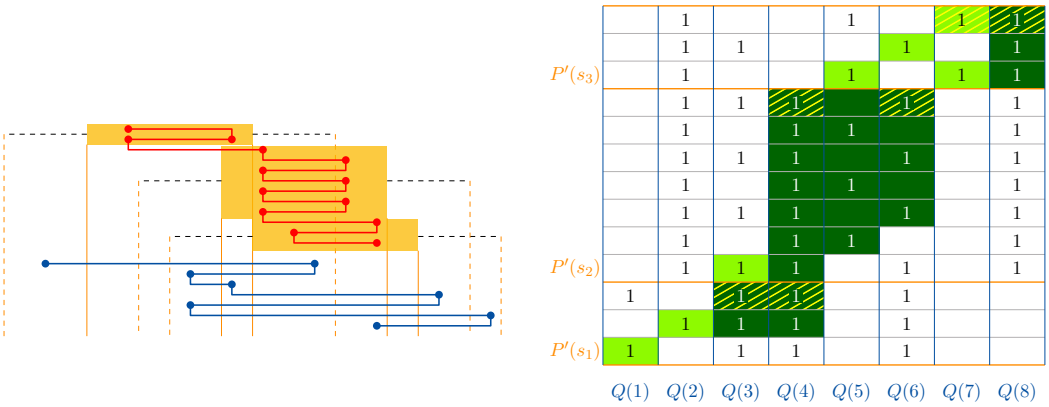


Figure 5 The compressed curve P' and the query curve Q are on the left. On the right is the free space matrix of P' and Q, where only 1-entries are added. The dark green cells visualize the parameter a and the light green cells the set $\{u - z \mid z \in A\}$. The hatched cells are the cells that have been in F_{pre} and F_{new}.

We prove this invariant via induction on j. For $j = 0$, the invariant clearly holds as $\{u_j - z \mid z \in A_j\} \cup \{a_j\} = \{\infty\}$. Now assume the invariant holds for $j-1$. Let $x \in \{0, \ldots, k_i - 1\}$. Then, by definition of the discrete Fréchet distance $(s_i + x, j)$ is reachable if and only if $|Q(j) - P(s_i + x)| \leq \frac{2}{3}\delta$ and $(s_i + x - 1, j)$, $(s_i + x, j - 1)$, or $(s_i + x - 1, j - 1)$ is reachable. We consider three different cases.

1. If $|Q(j) - p_i| > 2\delta$, then $(s_i + x, j)$ is not reachable, since $|Q(j) - P(s_i + x)| > \frac{2}{3}\delta$. By line 5 of Algorithm 2, the invariant holds.
2. If $|Q(j) - p_i| \leq \delta$, then $|Q(j) - P(s_i + x)| \leq \frac{2}{3}\delta$ for any $x \in \{0, \ldots, k_i - 1\}$. Further, observe that if $(s_i + y, j)$ is reachable for $y \in \{0, \ldots, k_i - 1\}$, then $(s_i + x, j)$ is also reachable for every $y \leq x \leq k_i - 1$. Define $y \in \{0, \ldots, k_i - 1\}$ to be the minimum such that $(s_i + y, j)$ is reachable. Hence, by line 13 it holds that the invariant holds if $a_j = y$. Unless $y = 0$ and $(s_i - 1, j)$ or $(s_i - 1, j - 1)$ is reachable, it must hold that $(s_i + y, j - 1)$ is reachable as otherwise y would not be the minimum. Further again since y is the minimum and by the invariant for $j - 1$, it holds that $y = \min\{u_{j-1} - z \mid z \in A_{j-1}\} \cup \{a_{j-1}\}$. Now, consider the special case that $y = 0$ and $(s_i - 1, j)$, or $(s_i - 1, j - 1)$ is reachable. By assumption, it holds that $(s_i - 1, j)$ (resp. $(s_i - 1, j - 1)$) is reachable if and only if $j \in F_{\text{pre}}$ (resp. $j - 1 \in F_{\text{pre}}$). Hence, a_j is set to y by line 9–12 and the invariant is true.
3. It remains the case that $\delta < |Q(j) - p_i| \leq 2\delta$.
   a. First, consider the case that $x \geq a_{j-1}$. Then since $(s_i + a_{j-1}, j - 1)$ is reachable by the induction hypothesis, it holds that $|Q(j-1) - p_i| \leq 2\delta$. Hence, $|Q(j-1) - P(s_i + x)| \leq \frac{2}{3}\delta$ or $|Q(j - 1) - P(s_i + x - 1)| \leq \frac{2}{3}\delta$. Further again by the induction hypothesis it holds that $(s_i + x, j - 1)$ or $(s_i + x - 1, j - 1)$ is reachable. Hence, by line 7–8 the invariant holds for $x \geq a_{j-1}$.
   b. Otherwise, it holds that $x < a_{j-1}$. Consider the special case that $x = 0$ and $(s_i + x, j)$ is reachable because $|Q(j) - P(s_i + x)| \leq \frac{2}{3}\delta$ and $(s_i - 1, j)$ or $(s_i - 1, j - 1)$ are reachable. Then by 9 and the assumption on F_{pre}, it holds that $u_j \in A_j$. Otherwise it holds that $(s_i + x, j)$ is reachable if and only if $(s_i + x, j - 1)$ or $(s_i + x - 1, j - 1)$ is reachable and $|Q(j) - p_i| \leq \frac{3}{2}\delta$. This is true because it holds that either $|Q(j) - P(s_i + x)| \leq \frac{2}{3}\delta$ or $|Q(j) - P(s_i + x - 1)| \leq \frac{2}{3}\delta$ since $\delta < |Q(j) - p_i| \leq 2\delta$. Hence, $x - 1 \in A_{j-1}$ or $x \in A_{j-1}$. Therefore, $A_{j-1} \neq [\,]$ and by the algorithm it holds that $\delta < |Q(j - 1) - p_i| \leq 2\delta$. If

$\min\{Q(j-1), Q(j)\} > p_i$ or $\max\{Q(j-1), Q(j)\} < p_i$, then $(s_i + x, j)$ is reachable if and only if $(s_i + x, j-1)$ is reachable by Observation 16. Otherwise $(s_i + x, j)$ is reachable if and only if $(s_i + x - 1, j - 1)$ is reachable. This is ensured by line 7–8.

This proves that the invariant is true for every j. Note that $(s_{i+1} - 1, j) = (s_i + k_i - 1, j)$. By the invariant it holds that $(s_i + k_i - 1, j)$ is reachable if and only if $|P'(s_{i+1} - 1) - Q(j)| \leq \frac{3}{2}\delta$ and $a_j < k_i$ or $u_j - k_i + 1 \in A_j$. Hence the lemma follows by line 15–16. ◀

▶ **Lemma 18.** *For any value $\delta \geq \delta_m$, we can decide whether $d_{dF}(P', Q) \leq \frac{3}{2}\delta$ in $\mathcal{O}(m\ell)$ time.*

Proof. We run Algorithm 2 and output $d_{\mathrm{dF}}(P', Q) \leq \frac{3}{2}\delta$ if and only if $m \in F_{\mathrm{new}}$ at the end of Algorithm 2. The running time of Algorithm 2 is in $\mathcal{O}(m\ell)$, since the array A is sorted and in every iteration of the inner for loop at most one point gets appended to A. By induction on the index i of s_i and Lemma 17, it holds that $(|P'|, m) = (s_{\ell+1} - 1, m)$ is reachable if and only if $d_{\mathrm{dF}}(P', Q) \leq \frac{3}{2}\delta$. ◀

It remains to optimize the δ value. First, we decide whether $d_{\mathrm{dF}}(P', Q) \leq \frac{3}{2}\delta_m$ or not using Algorithm 2. If $d_{\mathrm{dF}}(P', Q) \leq \frac{3}{2}\delta_m$, then $d_{\mathrm{dF}}(P, Q) \in [\delta_m, 2\delta_m]$ by Lemma 15 and δ_m is a 2-approximation for $d_{\mathrm{dF}}(P, Q)$. Otherwise $d_{\mathrm{dF}}(P', Q) > \frac{3}{2}\delta_m$. In this case, we use the observation that the discrete Fréchet distance between P' and Q gets realizes by a point to point distance between a vertex of P' and a vertex of Q. Hence,

$$d_{\mathrm{dF}}(P', Q) \in \{|(p_i \pm \delta/2) - Q(j)| \mid i \in \{1, \ldots, \ell\}, j \in \{1, \ldots, m\}\}.$$

We sort these values in $\mathcal{O}(m^2 \log m)$ time and search over the ones that are greater $\frac{3}{2}\delta_m$. Using Algorithm 2, we compute in $\mathcal{O}(m^2 \log m)$ time $\frac{3}{2}\delta = d_{\mathrm{dF}}(P', Q)$. Then, it holds that $d_{\mathrm{dF}}(P, Q) \in [\delta, 2\delta]$ by Lemma 15. Therefore, we get the following theorem:

▶ **Theorem 19.** *For a given value m, there exists a data structure that preprocesses a 1-dimensional curve P of complexity n in $\mathcal{O}(n \log n)$ time using $\mathcal{O}(m)$ space such that for any 1-dimensional query curve Q of complexity at most m, we can approximate the discrete Fréchet distance between P and Q within a factor of 2 in $\mathcal{O}(m^2 \log m)$ time.*

The next corollary is a direct consequence of the theorem above.

▶ **Corollary 20.** *Given two 1-dimensional curves P and Q of complexity n and m with $n \geq m$. There exists a 2-approximation algorithm to compute the discrete Fréchet distance between P and Q in $\mathcal{O}(n \log n + m^2 \log m)$ time.*

By Theorem 9, it follows that there does not exist an algorithm using $\mathcal{O}(n \log n + m^2 \log m)$ time with a better approximation factor. Further, the lower bound in [12] shows the almost optimality of the running time.

5 $(3 + \varepsilon)$-approximation Algorithm for the Fréchet distance

In this section, we give a $(3+\varepsilon)$-approximation algorithm to compute the continuous/discrete Fréchet distance between two curves P and Q of complexity n resp. m with $m \leq n$. We describe this section for the continuous Fréchet distance, but everything works analogously for the discrete Fréchet distance as well. We begin with the decision variant for δ. In the first step, using the curve Q, we compute a curve P' of complexity at most $2m$ such that $d_{\mathrm{F}}(P, P') \leq \delta$ or decide that $d_{\mathrm{F}}(P, Q) > \delta$ in linear time.

▶ **Lemma 21.** *Algorithm 3 computes a curve P' of complexity at most $2m$ with $d_F(P, P') \leq \delta$ or returns $d_F(P, Q) > \delta$ in $\mathcal{O}(n)$ time.*

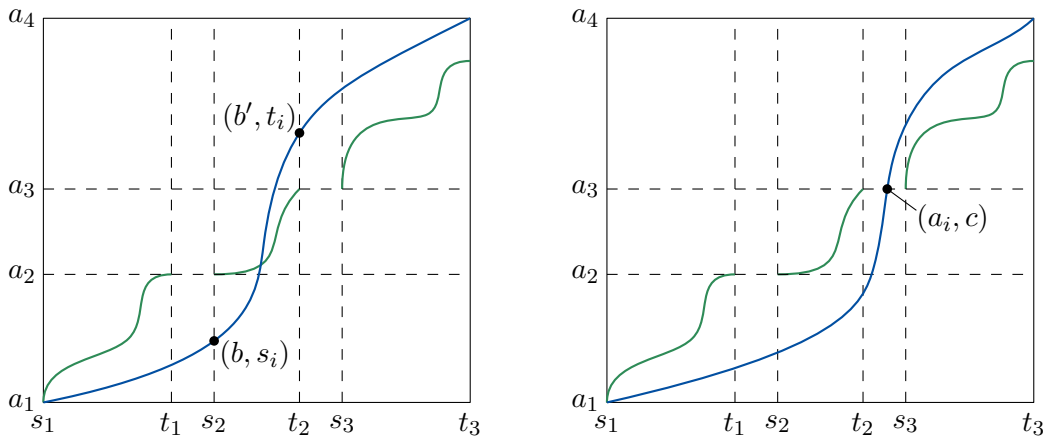


Figure 6 The green curve is a matching between the curves $P[a_i, a_{i+1}]$ and $Q[s_i, t_i]$ computed in Algorithm 3. The blue curves visualize the matching used in the proof of Lemma 21.

Proof. In the i-th iteration of the while-loop, it holds that $s_i \geq i$ by line 3 of Algorithm 3. Further, it holds that $a_i \leq a_{i+1}$ for every i and line 6 has running time $\mathcal{O}(a_{i+1} - a_i)$ as $\overline{Q(s_i)Q(t_i)} \subset \overline{Q(s_i)Q(\lfloor s_i + 1 \rfloor)}$. Hence, the running time follows. If a curve P' is returned, it holds that $d_{\mathrm{F}}(P, P') \leq \delta$ by line 6 and Observation 2. The complexity of P' is at most $2m$ since there are at most m iterations of the while loop. It remains to show correctness for the case when $d_{\mathrm{F}}(P, Q) > \delta$ is returned. Assume that $d_{\mathrm{F}}(P, Q) > \delta$ is returned and in fact it holds $d_{\mathrm{F}}(P, Q) \leq \delta$. Let M be the matching between P and Q that realizes the Fréchet distance. We consider two cases (see Figure 6). In the first case, there exist b, b' and an index i such that $(b, s_i), (b', t_i) \in M$ with $b \leq a_i$ and $b' > a_{i+1}$. This contradicts line 6 of the algorithm. Otherwise, there must exist an index i and a parameter c such that $(a_i, c) \in M$ with $t_{i-1} \leq c < s_i$. This contradicts line 3. Hence, it follows that $d_{\mathrm{F}}(P, Q) > \delta$. ◀

Algorithm 3 Simplification of P depending on Q.

```
1  s_1 ← 1, a_1 ← 1, i ← 1
2  while True do
3      s_i ← min{s ≥ ⌊s_{i−1} + 1⌋ | ‖P(a_i) − Q(s)‖ ≤ δ} ∪ {∞}
4      if s_i = ∞ then
5          Return d_F(P, Q) > δ
6      Find largest a_{i+1} such that ∃ t_i ≤ ⌊s_i + 1⌋ with d_F(P[a_i, a_{i+1}], Q(s_i)Q(t_i)) ≤ δ
7      if a_{i+1} = n then
8          Return P' = ⟨Q(s_1), Q(t_1), Q(s_2), Q(t_2), ..., Q(s_i), Q(t_i)⟩
9      i ← i + 1
```

Lemma 21 can be easily adapted such that it holds for the discrete Fréchet distance. Given P', we decide whether $d_{\mathrm{F}}(P', Q) \leq 2\delta$ in $\mathcal{O}(m^2)$ time [3] or slightly faster [15]. If $d_{\mathrm{F}}(P', Q) \leq 2\delta$, then $d_{\mathrm{F}}(P, Q) \leq d_{\mathrm{F}}(P', Q) + d_{\mathrm{F}}(P, P') \leq 3\delta$. Otherwise, it holds that $d_{\mathrm{F}}(P, Q) \geq d_{\mathrm{F}}(P', Q) - d_{\mathrm{F}}(P, P') > \delta$. Hence, we obtain the following lemma.

▶ **Lemma 22.** *Given two curves P and Q of complexity n and m in any metric space M. Assume that we can compute distances in M in constant time. We can 3-approximate the decision variant of the continuous/discrete Fréchet distance in $\mathcal{O}(n + m^2)$ time.*

For curves in the Euclidean space, Colombe and Fox (Theorem 11 of [18]) show how to transform a decision algorithm into an approximate optimization algorithm increasing the running time by a $\mathcal{O}(\log(n/\varepsilon))$ factor. Since all L_p-norms on $\mathbb{R}^d$ are equivalent and Lemma 22 holds for all L_p-norms, we can adopt their approach such that we can use it for all L_p-norms.

▶ **Theorem 23.** *There exist a $(3+\varepsilon)$-approximation algorithm for the continuous/discrete Fréchet distance using $\mathcal{O}((n+m^2)\log(n/\varepsilon))$ time.*

References

1. Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM Journal on Computing*, 43(2):429–449, 2014. doi:10.1137/130920526.
2. Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42(3-4):203–219, 2005. doi:10.1007/S00453-005-1165-Y.
3. Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995. doi:10.1142/S0218195995000064.
4. Sergey Bereg, Minghui Jiang, Wencheng Wang, Boting Yang, and Binhai Zhu. Simplifying 3d polygonal chains under the discrete Fréchet distance. In Eduardo Sany Laber, Claudson F. Bornstein, Loana Tito Nogueira, and Luérbio Faria, editors, *LATIN 2008: Theoretical Informatics, 8th Latin American Symposium, Búzios, Brazil, April 7-11, 2008, Proceedings*, volume 4957 of *Lecture Notes in Computer Science*, pages 630–641. Springer, 2008. doi:10.1007/978-3-540-78773-0_54.
5. Lotte Blank and Anne Driemel. A faster algorithm for the Fréchet distance in 1d for the imbalanced case. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 28:1–28:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.28.
6. Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless seth fails. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 661–670. IEEE, 2014. doi:10.1109/FOCS.2014.76.
7. Karl Bringmann and Bhaskar Ray Chaudhury. Polyline simplification has cubic complexity. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry, SoCG 2019, June 18-21, 2019, Portland, Oregon, USA*, volume 129 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.SOCG.2019.18.
8. Karl Bringmann, Anne Driemel, André Nusser, and Ioannis Psarros. Tight bounds for approximate near neighbor searching for time series under the Fréchet distance. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 517–550. SIAM, 2022. doi:10.1137/1.9781611977073.25.
9. Karl Bringmann and Marvin Künnemann. Multivariate fine-grained complexity of longest common subsequence. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1216–1235. SIAM, 2018. doi:10.1137/1.9781611975031.79.
10. Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *Journal of Computational Geometry*, 7(2):46–76, 2016. doi:10.20382/JOCG.V7I2A4.
11. Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four soviets walk the dog: Improved bounds for computing the Fréchet distance. *Discrete Comput. Geom.*, 58(1):180–216, July 2017. doi:10.1007/s00454-017-9878-7.

12 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. SETH says: Weak Fréchet distance is faster, but only if it is continuous and in one dimension. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2887–2901. SIAM, 2019. doi:10.1137/1.9781611975482.179.

13 Siu-Wing Cheng and Haoqiang Huang. Curve simplification and clustering under Fréchet distance. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1414–1432. SIAM, 2023. doi:10.1137/1.9781611977554.CH51.

14 Siu-Wing Cheng and Haoqiang Huang. Solving Fréchet distance problems by algebraic geometric methods. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4502–4513. SIAM, 2024. doi:10.1137/1.9781611977912.158.

15 Siu-Wing Cheng and Haoqiang Huang. Fréchet distance in subquadratic time. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 5100–5113. SIAM, 2025. doi:10.1137/1.9781611978322.173.

16 Siu-Wing Cheng, Haoqiang Huang, and Le Jiang. Simplification of trajectory streams. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry, SoCG 2025, June 23-27, 2025, Kanazawa, Japan*, volume 332 of *LIPIcs*, pages 34:1–34:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SOCG.2025.34.

17 Siu-Wing Cheng, Haoqiang Huang, and Shuo Zhang. Constant approximation of Fréchet distance in strongly subquadratic time. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 2329–2340. ACM, 2025. doi:10.1145/3717823.3718157.

18 Connor Colombe and Kyle Fox. Approximating the (continuous) Fréchet distance. In Kevin Buchin and Éric Colin de Verdière, editors, *37th International Symposium on Computational Geometry, SoCG 2021, June 7-11, 2021, Buffalo, NY, USA (Virtual Conference)*, volume 189 of *LIPIcs*, pages 26:1–26:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.SOCG.2021.26.

19 Anne Driemel and Sariel Har-Peled. Jaywalking your dog: Computing the Fréchet distance with shortcuts. *SIAM J. Comput.*, 42(5):1830–1866, 2013. doi:10.1137/120865112.

20 Anne Driemel, Ioannis Psarros, and Melanie Schmidt. Sublinear data structures for short Fréchet queries. *CoRR*, abs/1907.04420, 2019. arXiv:1907.04420.

21 Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance, 1994.

22 Arnold Filtser and Omrit Filtser. Static and streaming data structures for Fréchet distance queries. *ACM Trans. Algorithms*, 19(4):39:1–39:36, 2023. doi:10.1145/3610227.

23 Greg N. Frederickson and Donald B. Johnson. Generalized selection and ranking: Sorted matrices. *SIAM J. Comput.*, 13(1):14–30, 1984. doi:10.1137/0213002.

24 Joachim Gudmundsson, Martin P. Seybold, and Sampson Wong. Map matching queries on realistic input graphs under the Fréchet distance. *ACM Trans. Algorithms*, 20(2):14, 2024. doi:10.1145/3643683.

25 Aviad Rubinstein. Hardness of approximate nearest neighbor search. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1260–1268. ACM, 2018. doi:10.1145/3188745.3188916.

26 Thijs van der Horst and Tim Ophelders. Faster Fréchet distance approximation through truncated smoothing. *CoRR*, abs/2401.14815, 2024. doi:10.48550/arXiv.2401.14815.

27 Thijs van der Horst, Marc J. van Kreveld, Tim Ophelders, and Bettina Speckmann. A subquadratic n^{ϵ}-approximation for the continuous Fréchet distance. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1759–1776. SIAM, 2023. doi:10.1137/1.9781611977554.CH67.

28 Marc J. van Kreveld, Maarten Löffler, and Lionov Wiratma. On optimal polyline simplification using the Hausdorff and Fréchet distance. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, volume 99 of *LIPIcs*, pages 56:1–56:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.SOCG.2018.56.

29 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/J.TCS.2005.09.023.

Product Structure and Treewidth of Hyperbolic Uniform Disk Graphs

Thomas Bläsius
Karlsruhe Institute of Technology, Germany

Emil Dohse
Karlsruhe Institute of Technology, Germany

Deborah Haun
Karlsruhe Institute of Technology, Germany

Laura Merker
Karlsruhe Institute of Technology, Germany

Abstract

Hyperbolic uniform disk graphs (HUDGs) are intersection graphs of disks with some radius r in the hyperbolic plane, where r may be constant or depend on the number of vertices in a family of HUDGs. We show that HUDGs with constant clique number do not admit *product structure*, i.e., that there is no constant c such that every such graph is a subgraph of $H \boxtimes P$ for some graph H of treewidth at most c. This justifies that HUDGs are described as not having a grid-like structure in the literature, and is in contrast to unit disk graphs in the Euclidean plane, whose grid-like structure is evident from the fact that they are subgraphs of the strong product of two paths and a clique of constant size [Dvořák et al., '21, MATRIX Annals]. By allowing H to be any graph of constant treewidth instead of a path-like graph, we reject the possibility of a grid-like structure not merely by the maximum degree (which is unbounded for HUDGs) but due to their global structure. We complement this by showing that for every (sub-)constant r, HUDGs admit product structure, whereas the typical hyperbolic behavior is observed if r grows with the number of vertices.

Our proof involves a family of n-vertex HUDGs with radius $\log n$ that has bounded clique number but unbounded treewidth, and one for which the ratio of treewidth and clique number is $\log n / \log \log n$. Up to a $\log \log n$ factor, this negatively answers a question raised by Bläsius et al. [SoCG '25] asking whether balanced separators of HUDGs with radius $\log n$ can be covered by less than $\log n$ cliques. Our results also imply that the local and layered tree-independence number of HUDGs are both unbounded, answering an open question of Dallard et al. [arXiv '25].

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases hyperbolic uniform disk graphs, product structure, treewidth

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.18

Related Version *Full Version*: `https://arxiv.org/abs/2603.18997` [2]

Funding Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Projektnummer 524989715.

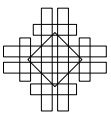
Acknowledgements We thank Torsten Ueckerdt and Marcus Wilhelm for fruitful discussions.

1 Introduction

In this paper, we investigate the structure of *hyperbolic uniform disk graphs (HUDGs)*, i.e., graphs G for which there is an r such that G is the intersection graph of disks with radius r in the hyperbolic plane. The structure of these graphs heavily depends on r, where r may be constant or a function depending on the number n of vertices for a family of HUDGs. In the literature [3, 5], HUDGs with very small r are described as almost Euclidean or grid-like, and as firmly hyperbolic or more hierarchical for larger r; see Figure 1. However, the state of the



© Thomas Bläsius, Emil Dohse, Deborah Haun, and Laura Merker;
licensed under Creative Commons License CC-BY 4.0



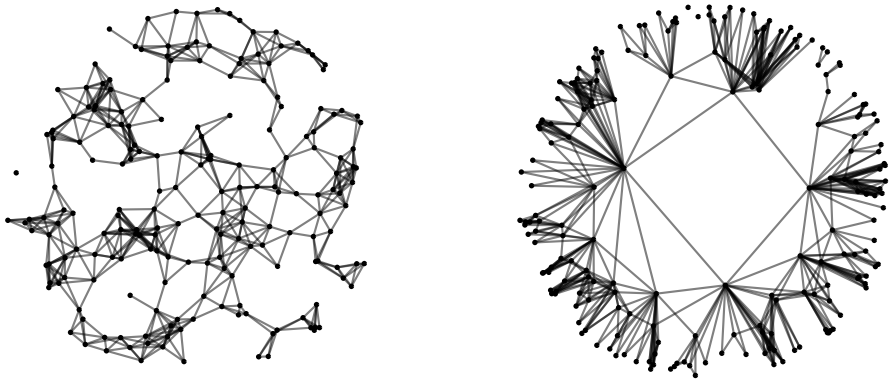


Figure 1 HUDGs with random vertex positions. Left: The radius r is so small (think of $1/\sqrt{n}$) that it is indistinguishable from the Euclidean setting. Right: The disk radius r is logarithmic in n.

art cannot fully explain in what sense the grid-like structure depends on r. We contribute to the understanding of the structure of HUDGs by formalizing when a graph has a grid-like structure and showing that this is indeed not satisfied by HUDGs, unless the radius is small. We do so by showing that HUDGs do not admit a so-called *product structure*, and thereby also deepen the understanding of this notion. Before stating our results precisely, we review what is already known on the structure of HUDGs and how it depends on the radius r.

We start with small radii and work our way up from there. Locally, the hyperbolic plane behaves like the Euclidean plane, which sets the expectation that very small r yield graphs similar to *Euclidean unit disk graphs (EUDGs)*, which are defined as intersection graphs of unit disks in the Euclidean plane. While there are always HUDGs that are not EUDGs independent of how small we choose r, every EUDG is in fact a HUDG if we choose r sufficiently small [3]. Sufficiently small here means that r shrinks with the number of vertices n. EUDGs [8] are a well studied graph class that allows for balanced separators that can be covered with $O(\sqrt{n})$ cliques [14]. We remark that balanced separators are arguably one of the main reasons why HUDGs, EUDGs, and more generally disk graphs, are interesting from an algorithmic perspective. For HUDGs, there are balanced separators that can be covered with $O((1+1/r)\log n)$ cliques [5]. Interestingly, this matches the $\sqrt{n}$ bound of the Euclidean case for $r \in \Theta(1/\sqrt{n})$ up to a $\log n$ factor and improves for larger r. In fact, if $r \in \Omega(1)$, this gives separators coverable with $\log n$ cliques. This gives a strong argument that the structure of HUDGs changes when going from sub-constant to at least constant r. This change in structure is also reflected in the fact that several problems can be solved more efficiently for constant r than in the Euclidean setting [5, 30]. Summarizing, there is strong evidence that the structure of HUDGs change significantly between sub-constant r and constant r.

This raises the question whether there is a similar structural difference between constant r and super-constant r. While the above separators do not continue to get smaller, there are indications that super-constant r makes a difference, and that in particular $r \in \Theta(\log n)$ is an interesting case. For HUDGs with $r \in \Theta(\log n)$, the independent set problem can be solved in polynomial time [5]. Moreover, HUDGs with $r \in \Theta(\log n)$ and random vertex positions are so-called hyperbolic random graphs [31]; a random graph model that is popular due to its resemblance to real-world networks in regards to properties like clustering, diameter, and degree distribution [22, 23, 26]. While this indicates a significant difference between constant and growing r, the state of the art does not make the structural differences explicit. The only structural insight we are aware of is that large stars, or more generally high-degree vertices with sparse neighborhoods, require r to grow logarithmically with the degree [3].

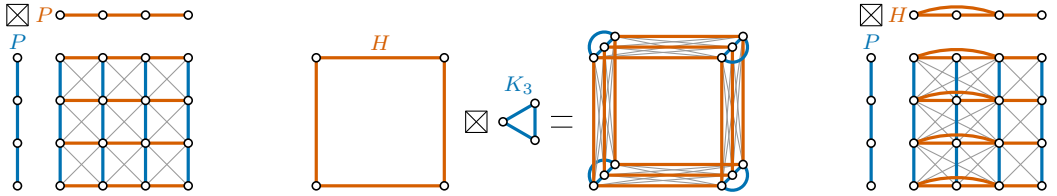


Figure 2 The strong product of two paths (left), of a graph H and a clique (middle), and a graph H with a path (right).

The main result of this paper is to show that every class of HUDGs with constant radius r has product structure, while there are HUDGs with growing radius that do not. This closes two gaps in the literature: First, it justifies why HUDGs with large radii should be considered *not* grid-like. And second, the contrast to HUDGs with constant radius shows a stark structural difference between constant radius compared to larger radii, say $\log n$.

Concerning the notion of grid-like structure, observe that $\sqrt{n} \times \sqrt{n}$-grids are EUDGs and thus also HUDGs for small radius. However, the grid-like structure of EUDGs goes far beyond merely containing grids. In fact, EUDGs are in general rather grid-like in the sense that there may be large cliques locally, but the global structure of every EUDG behaves like (the subgraph of) a grid. This notion of being "grid-like" can be formalized using the *strong product* of graphs, which we introduce formally in Section 2. For now, it suffices to know that the strong product $P_a \boxtimes P_b$ of two paths on a and b vertices, respectively, is the $a \times b$-grid with diagonals. Moreover, the strong product $H \boxtimes K_k$ of a graph H with the complete graph K_k expands each vertex of H to a k-clique as shown in Figure 2. Dvořák, Huynh, Joret, Liu, and Wood [20] show that every EUDG G is a subgraph of $P \boxtimes P \boxtimes K_k$, where P is a path and k is linear in the clique number ω of G. Equivalently, one can partition the vertices of a EUDG into sets of size $O(\omega)$ so that contracting each set into a single vertex yields a subgraph H of a grid (with diagonals), yielding a product of the form $H \boxtimes K_k$.

More generally, the concept of product structure allows us to view the graphs of a graph class as subgraphs of the product of simpler graphs; usually a graph with small treewidth and a path. We say a graph class $\mathcal{G}$ admits *product structure* if for every $G \in \mathcal{G}$, we have $G \subseteq H'' \boxtimes P$ for some graph H'' of constant treewidth and some path P. In our proofs, we show a stronger statement, namely that the graphs we consider are subgraphs of $H \boxtimes K_k$ for some hyperbolic tiling H. Since planar graphs, which include hyperbolic tilings, admit product structure, this implies $H \subseteq H' \boxtimes P \boxtimes K_3$ for some graph H of constant treewidth [16,17], so with $H'' = H' \boxtimes K_3 \boxtimes K_k$ we obtain a product of the form $H'' \boxtimes P$. Note that the above mentioned result on EUDGs can be viewed in this framework by choosing $H = P \boxtimes P$. We remark that replacing one of the paths in the product $P \boxtimes P \boxtimes K_k$ by a graph H', enables graphs that go beyond Euclidean square-grids. In particular, it allows for high-degree vertices. That is, rejecting a grid-like structure via product structure is significantly stronger than showing that a graph class does not allow for a decomposition of the form $P \boxtimes P \boxtimes K_k$. To finish the discussion on grid-like structure, let us mention that "having product structure" does not need to be seen as a binary decision. For this, the *row-treewidth* of a graph G is the smallest t such that $G \subseteq H'' \boxtimes P$ for some graph H'' of treewidth t and some path P. The row-treewidth of a graph class is the maximum

row-treewidth among all contained graphs, possibly as a function depending on the number of vertices if it is unbounded. Now, among graph classes admitting product structure, we consider a class to have a stronger grid-like structure if the row-treewidth is small.

Our results that some families of HUDGs admit product structure while others do not, make HUDGs also interesting from the product structure perspective. Product structure has proved to be a useful tool for many graph-theoretic (see [28] for a survey) and algorithmic [1,6] applications. While naturally, the literature concentrates on graph classes that admit product structure [7, 15, 16, 19, 27, 28], it is also interesting to find graph classes that do not admit product structure, for an other reason than the presence of large cliques. Interestingly, many known constructions are geometric intersection graphs, namely different kinds of string graphs with constant clique number [29, 32]. By showing that HUDGs do not admit product structure, we contribute to an improved understanding of the limitations of this notion.

1.1 Main Results and Discussion

The following theorem states our main result that HUDGs do not admit product structure, even when restricted to constant clique size, rejecting the possibility of a global grid-structure.

▶ **Theorem 1.** *Hyperbolic uniform disk graphs with constant clique number do not admit product structure.*

Our proof of Theorem 1 requires the radius to grow with the graph size. This is indeed necessary as HUDGs with constant or sub-constant disk radius do admit product structure.

▶ **Theorem 2.** *Every family of hyperbolic uniform disk graphs with clique number and disk radius in $O(1)$ admits product structure. In contrast, for every super-constant r, there are families of hyperbolic uniform disk graphs with constant clique number and disk radius in $\Theta(r)$ not admitting product structure.*

This confirms the observations of previous results [3, 5] that HUDGs become similar to the Euclidean case for small radii, whereas they differ significantly for large radii. As having product structure in some sense restricts the complexity of the neighborhood of an individual vertex, this in particular indicates that HUDGs become more complex (and for some applications more interesting) when allowing the radius to grow with the graph size. We show the first part of Theorem 2 in Section 4 and the second in Section 3.

Bounds on the Row-Treewidth. We actually prove more fine-grained results that go beyond the binary question of having product structure. Recall that a graph class admits product structure if and only if it has bounded row-treewidth, i.e., a super-constant lower bound and a constant upper bound on the row-treewidth disprove and prove product structure, respectively. Our lower and upper bounds on the row-treewidth depending on clique number ω and disk radius r are shown in Table 1. The first two rows show our lower bounds that grow with r (as long as $r \in O(\log n)$), where we have the two cases $\omega \in O(1)$ and $\omega \in O(\log \log n)$. The first case is most relevant for product structure as growing clique size immediately rejects product structure, whereas the second case gives the stronger lower bound. Then, the third row gives our upper bound, whereas the last row is from the literature and is discussed at the end of this section. Before doing so, we discuss each result in Table 1 in more detail.

The first lower bound in Table 1 assumes constant clique number $\omega \in O(1)$. Note that there are two regimes for the radius r. For radii smaller than $\log n$, the lower bound grows with increasing radius until it reaches a row-treewidth of $\Omega(\log \log n)$ for radius $r \in \Theta(\log n)$. For larger radii, the lower bound stops to grow with r. We note that any super-constant

Table 1 Overview of lower and upper bounds for the row-treewidth. The lower bounds mean that for every radius r in the given regime, there exists a family of HUDGs with radius $\Theta(r)$ and the given bounds on the clique number ω and the row-treewidth. The upper bounds hold for all HUDGs with radius r and clique number ω (both of which may depend on the graph size n).

ω	row-treewidth		comment
	$r \in O(\log n)$	$r \in \Omega(\log n)$	
$O(1)$	$\Omega(\log r)$	$\Omega(\log \log n)$	Cor. 15 and 16; implies Thm. 1 and 2
$O(\log \log n)$	$\Omega(r)$	$\Omega(\log n)$	Cor. 15 and 16
variable	$O(\omega \cdot 3^{8r})$		Thm. 17; implies Thm. 2 for $r, \omega \in O(1)$
variable	$O(\omega(1+\frac{1}{r})\log n)$		follows from treewidth bound [5]

radius already implies super-constant row-treewidth and thus refutes product structure, yielding Theorem 1 and the second part of Theorem 2. That is, the specific lower bounds are significantly stronger than Theorem 1 as they refute product structure for every growing radius and not only for the class of all HUDGs. For the lower bound given in the next row of Table 1, we allow slightly growing clique number $\omega \in O(\log \log n)$, which yields exponentially larger lower bounds for the row-treewidth compared to the first row. We remark that all our lower bounds are linear in the size of the largest clique-minor, known as the Hadwiger number.

Turning to our upper bound (bottom part of Table 1), observe that it only depends on ω and r. Assuming both to be constant directly implies the first part of Theorem 2. Moreover, for growing clique number ω (but constant r), this is the best upper bound one can hope for, as the clique number is a lower bound on the row-treewidth. Concerning growing radius, note that the radius r contributes exponentially to the row-treewidth. While it might be possible to improve this upper bound, we note that our lower bounds for growing radius imply that there has to be a dependence on r. We remark that we strengthen our upper bound in terms of product structure. For this, recall that the upper bound on the row-treewidth means that every HUDG with radius r is a subgraph of $H'' \boxtimes P$ for some graph H'' of treewidth $O(\omega \cdot 3^{8r})$. We show the stronger statement that for every HUDG G with radius r and clique number ω, we have $G \subseteq H \boxtimes K_k \subseteq H' \boxtimes P \boxtimes K_k$, for some hyperbolic tiling H, some graph H' of treewidth 3, some path P, and $k \in O(\omega \cdot 3^{8r})$. Our proof is similar to the Euclidean case [20], but instead of the Euclidean square-grid, we construct a suitable tiling of the hyperbolic plane. Interestingly, the tiling depends on r and becomes more hyperbolic for large r and more Euclidean for small r. With our tiling, we also contribute to the rich field of constructing hyperbolic tilings with various favorable properties [9, 13, 33].

▶ **Theorem 3.** *For each $r > 0$ there is an (irregular) tiling of the hyperbolic plane such that*
- *every tile can be covered by $O(3^{8r})$ disks of radius r and*
- *each two points with distance at most $2r$ lie in the same tile or in two adjacent tiles.*

The last upper bound in Table 1 directly follows from the literature, which we discuss next.

Connections to the State of the Art. Our lower bounds are based on the fact that the row-treewidth is asymptotically lower-bounded by the treewidth in the neighborhood of a vertex [18]. Notably, the neighborhoods of HUDGs are called *strongly hyperbolic uniform disk graphs (SHUDGs)* and are an interesting graph class in their own right [3]. In fact, the much studied hyperbolic random graphs are SHUDGs with random vertex positions [31]. The core of our lower bounds for the row-treewidth of HUDGs are the following bounds on the treewidth of SHUDGs.

▶ **Theorem 4.** *There are families of n-vertex strongly hyperbolic uniform disk graphs with radius $\Theta(\log n)$ and*

- *clique number $O(\log\log n)$ and treewidth $\Omega(\log n)$*
- *clique number $O(1)$ and treewidth $\Omega(\log\log n)$.*

To connect this to the results by Bläsius, von der Heydt, Kisfaludi-Bak, Wilhelm, and Van Wordragen [5], recall that they show that n-vertex HUDGs with radius r have balanced separators that can be covered with $(1+1/r)\log n$ cliques. This implies that the treewidth is $O(\omega(1+1/r)\log n)$ [21], which is an upper bound for the row-treewidth (which is why we list it in Table 1). For r at least constant, this gives separators coverable with $O(\log n)$ cliques, i.e., treewidth $O(\omega\log n)$. The authors of [5] raise the question whether this can be improved, in particular for increasing radius r. With Theorem 4, we give a negative answer to this, up to a $\log\log n$ factor: The family with clique number $\omega\in O(\log\log n)$ has treewidth $\Omega(\log n)=\Omega(\omega\log n/\log\log n)$.

In addition, Theorem 4 provides the first known example of a family of SHUDGs with constant clique number but unbounded treewidth. Graph classes with this property are also called *not* (tw,ω)*-bounded*. By Dallard, Milanič, and Štorgel [12], graph classes with bounded tree-independence number are (tw,ω)-bounded, so our constructed family of SHUDGs has unbounded tree-independence number. Since SHUDGs are neighborhoods of vertices in HUDGs, it follows that the local tree-independence number of HUDGs is unbounded, which in turn implies that their layered tree-independence number is unbounded [24]. This gives a negative answer to an open question of Dallard, Milanič, Munaro, and Yang [11].

▶ **Corollary 5.** *The local tree-independence number and the layered tree-independence number of hyperbolic uniform disk graphs are unbounded.*

Note that for these two answered open questions, the first comes from the HUDG side and the other is more related to product structure. This underscores that HUDGs and product structure form an interesting combination.

2 Preliminaries

We assume familiarity with some graph-theoretic terminology; also see the full version [2].

2.1 Product Structure

Separators and Treewidth. The *treewidth* $\mathrm{tw}(G)$ of a graph G is linearly tied to the *separation number*, i.e., the smallest integer s such that every subgraph of G has a balanced separator of size s [21], [10, Lemma 7.20]. If G has a clique-minor of size k, then $\mathrm{tw}(G)\geq k-1$.

Strong Product. For two graphs G and H, the *strong product* $G\boxtimes H$ is defined as follows; also see Figure 3. $G\boxtimes H$ has vertex set $V(G\boxtimes H)=V(G)\times V(H)$, i.e., every vertex $u=(u_G,u_H)$ of $G\boxtimes H$ corresponds to a vertex u_G in G and to a vertex u_H in H. Two vertices $u=(u_G,u_H)$ and $v=(v_G,v_H)$ are adjacent in $G\boxtimes H$ if one of the following is true.

- u and v are the same vertex in G and adjacent in H, i.e., $u_G=v_G$ and $u_Hv_H\in E(H)$.
- u and v are the same vertex in H and adjacent in G, i.e., $u_H=v_H$ and $u_Gv_G\in E(G)$.
- u and v are adjacent in G and H, i.e., $u_Gv_G\in E(G)$ and $u_Hv_H\in E(H)$.

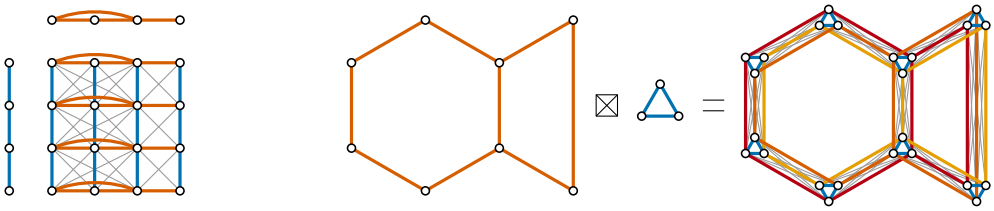


Figure 3 Left: Strong product of a graph H (red) with a path P (blue). The three different types of edges are color coded in red (edge in H, same vertex in P), blue (edge in P, same vertex in H), and gray (edge in H and P). Right: Strong product of a graph (red) with a K_3 (blue).

Row-Treewidth and Product Structure. We are usually interested in products of the form $H \boxtimes P$ where H has small treewidth and P is a path (of arbitrary length). Assume G is a subgraph of such a product $G \subseteq H \boxtimes P$ and assume that H is chosen such that $\text{tw}(H)$ is minimum with this property. Then, we call $\text{tw}(H)$ the *row-treewidth* of G. We say that a graph family $\mathcal{G}$ has *product structure* if the graphs in $\mathcal{G}$ have row-treewidth $O(1)$, i.e., there exists a constant that upper-bounds the row-treewidth of every graph $G \in \mathcal{G}$.

For products of the form $G = H \boxtimes K_k$, think of G as being obtained as follows. Start with H and replace each vertex with a k-clique and each edge with a complete bipartite graph between the corresponding k-cliques. The treewidth of $H \boxtimes K_k$ is $(\text{tw}(H) + 1) \cdot k - 1$.

2.2 Hyperbolic Geometry and Tilings

While this paper is about graphs in the hyperbolic plane, most of it can be understood with only little knowledge of hyperbolic geometry. We thus only provide a brief introduction. Formally, hyperbolic geometry is obtained from Euclidean geometry by replacing the parallel axiom: Instead of having exactly one parallel to a line through some given point not on the line, there are infinitely many, where two lines are considered parallel if they do not intersect. Thus, many concepts familiar from Euclidean geometry still apply as long as the parallel axiom is not involved, e.g., definitions of distances, angles, disks, (regular) polygons, and congruency are unchanged and the triangle inequality holds. However, some geometric laws like trigonometric formulas work differently, which we only introduce when needed; see [25].

Basic Properties. Our illustrations sometimes use the Poincaré disk model, which maps the hyperbolic plane into a unit disk of the Euclidean plane. However, we do not need any knowledge about this model and the illustrations should be helpful regardless. The core property of the hyperbolic plane relevant for this paper is the following. The hyperbolic plane expands exponentially, while at the same time behaving like the Euclidean plane locally. To concretize, consider a disk D of radius r. If r is small, then the area and circumference of D are quadratic and linear in r, respectively, like in the Euclidean plane. If the radius r increases, the area and circumference both grow exponentially like $\Theta(e^r)$. This has two interesting effects. First, the hyperbolic plane has more space. This becomes more pronounced at bigger scales, i.e., when we look at a large region. Secondly, in Euclidean geometry one can scale objects to change their size while keeping them structurally the same, e.g., think of similar triangles. In the hyperbolic plane, however, the scale makes a big structural difference, and there is no scaling operation that changes all distances by the same factor.

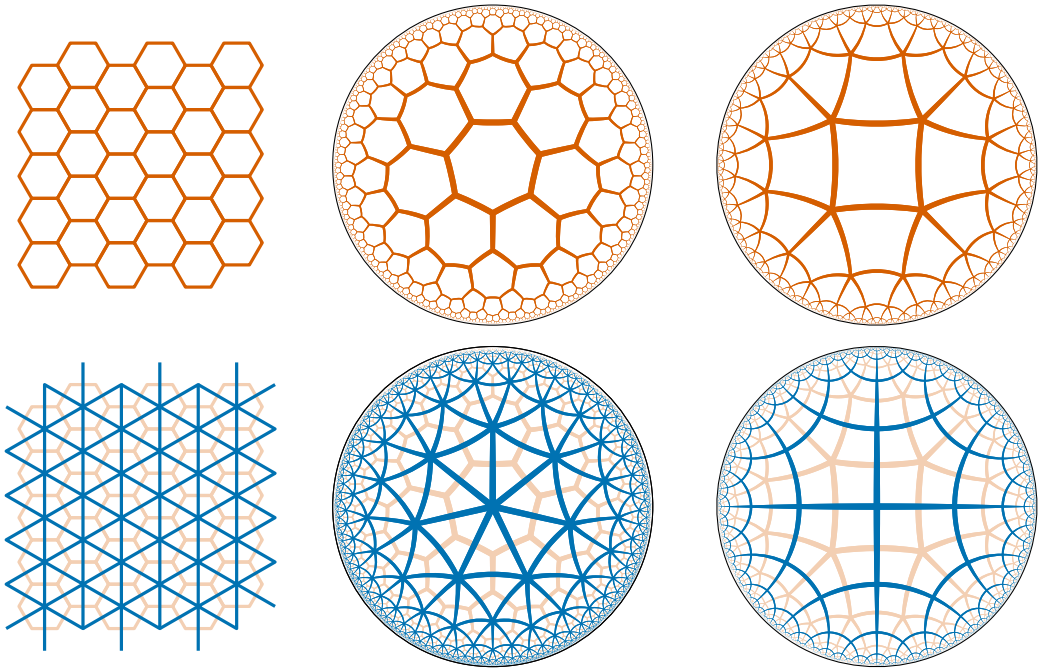


Figure 4 Regular tilings (top) together with their duals (bottom). From left to right: A Euclidean $\{6,3\}$ tiling, a hyperbolic $\{7,3\}$ tiling, and a hyperbolic $\{4,5\}$-tiling. While the tiles seem to get smaller towards the boundary of the Poincaré disk, they are all congruent.

Hyperbolic Tilings. A *regular tiling* is a partition of the plane into congruent regular polygons; see Figure 4. The Euclidean plane can be tiled with triangles, squares, and hexagons. These tilings have the Schläfli symbols $\{3,6\}$, $\{4,4\}$, and $\{6,3\}$, respectively, where $\{p,q\}$ indicates that each tile is a p-gon and q such p-gons meet at each corner. For any p and q with $\frac{1}{p}+\frac{1}{q}<\frac{1}{2}$, we obtain a regular tiling of the hyperbolic plane. While Euclidean tilings can be scaled arbitrarily, the size of the tiles in a hyperbolic $\{p,q\}$-tiling is determined by p and q and can be derived from basic trigonometry. Observe in Figure 4 that the exponential expansion of the hyperbolic plane is reflected in the tilings, which can be seen as a discretization of the plane. We also consider (irregular) tilings, where the tiles do not need to be congruent. We usually interpret tilings as infinite[1] plane graphs. Then the corners are the vertices, the sides are the edges, and the p-gons are the *faces*. The *dual* of such a plane graph has one vertex per face and two faces are adjacent if they share an edge.

2.3 Geometric Graphs

A graph G is a *(Euclidean) unit disk graph (EUDG)* if it is the intersection graph of disks of radius 1 in the Euclidean plane. We also interpret the disk centers as the *positions* of the corresponding vertices and use terms like *distance*, i.e., two vertices are connected if and only if their distance is at most 2. We say that G is a *hyperbolic uniform disk graph (HUDG)* if there exists a *disk radius* r such that G is the intersection graph of hyperbolic disks of radius r. The radius r can depend on the graph and makes an important difference.

[1] Throughout the paper, we implicitly assume that all considered graphs are finite. Tilings are the only exception. But also for tilings, we are usually only interested in a finite subgraph.

We are usually interested in the asymptotics of r with respect to the graph size n. We say that a HUDG G *has disk radius* r if it is the intersection graph of hyperbolic disks of radius r. Moreover, we say that a family $\mathcal{G}$ of HUDGs *has disk radius* $r(n)$ if every $G \in \mathcal{G}$ with n vertices has disk radius $r(n)$. This lets us use asymptotics, i.e., $\mathcal{G}$ *has disk radius* $\Theta(r(n))$ if there exist constants c_1, c_2 such that every sufficiently large n-vertex graph $G \in \mathcal{G}$ has disk radius r_G for $c_1 r(n) \leq r_G \leq c_2 r(n)$.[2]

A graph is a *strongly hyperbolic uniform disk graph (SHUDG)* [3] if it is a HUDG with disk radius r such that all vertices have distance at most $2r$ from the origin. This means that a vertex placed at the origin would be universal, i.e., adjacent to all other vertices of the graph. Note that the neighborhood of every vertex in a HUDG induces an SHUDG. Studying SHUDGs is in some sense similar to studying families of HUDGs with large disk radius: SHUDGs with bounded average degree have disk radius $\Omega(\log n)$ [3].

3 HUDGs Do Not Admit Product Structure

In this section, we show that HUDGs with constant clique number do not admit product structure (Theorem 1). While our construction initially has disk radius $\Theta(\log n)$, we extend it to arbitrary super-constant disk radii to prove the second part of Theorem 2. This is tight as we show in Section 4 that every family of HUDGs with disk radius in $O(1)$ and constant clique number has indeed product structure. Similarly, we provide families of HUDGs for the lower bounds in the second row of Table 1 for every super-constant disk radius. These results are summarized in Corollaries 15 and 16 at the end of the section.

To reject product structure, we use that, asymptotically, the treewidth in the neighborhood of some vertex is a lower bound on the row-treewidth [16, 18]. Since SHUDGs describe the neighborhoods of vertices in a HUDG, we aim for a family of SHUDGs with constant clique number but unbounded treewidth to prove Theorem 1 (see the full version [2] for details).

▶ **Theorem 6.** *There is a family of n-vertex SHUDGs with disk radius in $\Theta(\log n)$, clique number in $O(\log \log n)$, and treewidth in $\Omega(\log n)$.*

We then identify an induced subgraph within each of the constructed SHUDGs to prove Theorem 1. In addition, Theorems 6 and 7 together prove Theorem 4.

▶ **Theorem 7.** *There is a family of n-vertex SHUDGs with disk radius in $\Theta(\log n)$, clique number in $\Theta(1)$, and treewidth in $\Omega(\log \log n)$.*

3.1 Proof of Theorem 6

For each integer $r \geq 2$, we construct an n-vertex SHUDG G_r by placing $n \in \Theta(2^r)$ vertices into the hyperbolic plane, which yields $r \in \Theta(\log n)$ for the disk radius as required by Theorems 6 and 7. First, we place a vertex at the origin which we call the *root*. Second, for $2 \leq k \leq r$, let A_k be a regular 2^k-gon with side lengths $2r$ whose center is at the origin. We align these polygons such that for each $k > 2$, every other edge of A_k has the same perpendicular bisector as some edge of A_{k-1}; see Figure 5. Now, let V_k denote the vertex set of A_k and let V_1 contain exactly the root. We say a vertex is in the *k-th level* if it is contained in V_k. The vertex set of G_r is defined as the union of all V_k, $k \in [r]$, and two vertices are adjacent if and only if their distance is at most $2r$. In particular, the sides of

[2] We note that this is the natural way to use asymptotics. We only make this explicit as asymptotics can be unintuitive in this context, e.g., "the class of all HUDGs with radius $\Theta(\log n)$" is not well-defined.

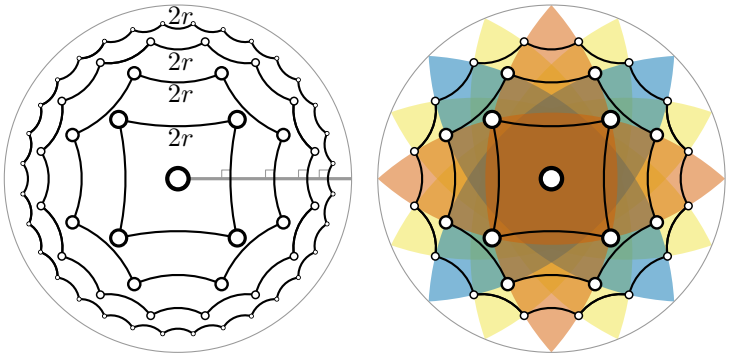


Figure 5 Left: Vertex positions of G_r constructed for Theorem 6. The k-th level, $k \geq 2$, induces a 2^k-cycle with consecutive vertices having angular distance $2\pi/2^k$. Right: Equivalent construction inspired by separators in [4], where vertices are placed at intersection points of hypercycles.

the polygons are also edges in G_r, and there are no further edges within a level as chords in a regular polygon are strictly longer than their sides. That is, each level induces a cycle, where the angular distance between any two adjacent vertices in level k is $2\pi/2^k$. Note that the number of vertices is indeed in $\Theta(2^r)$. In the full version [2, Lemma 11] we show that all vertices are adjacent to the root, i.e., G_r is a SHUDG, as required. Without going into detail, we note that the above construction is inspired by an analysis in [4]; see Figure 5.

To bound the clique number and treewidth of G_r, we count the neighbors in each level.

▶ **Lemma 8.** *For every $2 \leq i \leq j \leq r$, every vertex of G_r in level i has at least two and at most $4\sqrt{2^{j-i}}$ neighbors in level j, where the upper bound is tight up to a constant factor.*

Before giving a proof sketch, let us discuss the implications to the structure of G_r. For this, consider a vertex v in some level i and the two closest vertices in level $i+1$, one to the left and one to the right. Now let T denote the tree induced by the four edges from the root to the first level and the edges to the two closest neighbors for each vertex in levels $2, \ldots, r-1$, see Figure 6. Observe that, apart from the root having degree 4, T is a spanning binary tree. That is, the number of descendants of some vertex grows with base 2 in T, whereas the number of neighbors in G_r grows with base $\sqrt{2}$. Hence, for a vertex in level i, there are far fewer neighbors than descendants in level j, provided $j - i$ is sufficiently large. This observation is key to bounding the clique number of G_r.

Proof sketch. Observe that for a vertex v_i in level i, the two closest vertices v_j^ℓ, v_j^r in level j are adjacent and have a very small angular distance to v_i (see Figure 6). Since v_j^ℓ, v_j^r are connected to the root, one can conclude that they also have edges to v_i.

For the upper bound, we use some trigonometric formulas to bound the maximum angular distance of two adjacent vertices with fixed distances from the root. Since we know from the construction how many vertices we have in each level, this gives us the number of neighbors, see again Figure 6. We refer to [2, Lemma 13] for a full proof. ◀

By the lower bound of Lemma 8, since each two levels are connected by edges, contracting each of the r levels yields an r-clique.

▶ **Corollary 9.** *The treewidth of G_r is at least $r \in \Omega(\log n)$, where n is the number of vertices.*

Next, the upper bound enables us to conclude the following lemma for triangles in G_r.

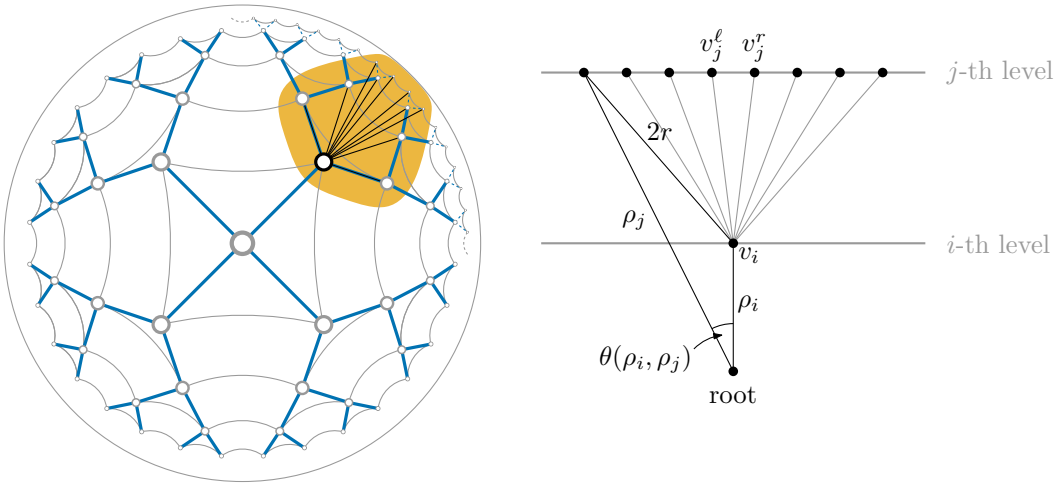


Figure 6 Left: A spanning tree (blue) guiding the structure of G_r, and the neighborhood of some vertex (black and orange). Note that the number of neighbors increases with the level, but slower than the number of descendants in the tree, whereas the angle in which a vertex has neighbors shrinks. Right: The number of neighbors a vertex v_i has in level j is determined by the angle $\theta(\rho_i, \rho_j)$ to the leftmost neighbor, where ρ_i and ρ_j are the radii of the two levels.

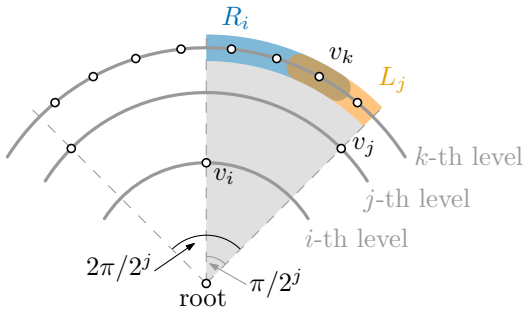


Figure 7 A triangle of G_r, where v_k lies both in the neighborhood of v_i (blue) and of v_j (orange). We conclude that the right half of the neighborhood of v_i plus the left half of the neighborhood of v_j is larger than the number of vertices in level k that lie between v_i and v_j (gray wedge).

▶ **Lemma 10.** *For every triangle in G_r with vertices v_i, v_j, and v_k on levels $2 \leq i \leq j \leq k$, respectively, we have $k - j \leq j - i + 6$.*

Proof sketch. We consider the number of neighbors v_i and v_j have in level k and compare the sum with the number of vertices in level k that lie between v_i and v_j (Figure 7). The first is $O(\sqrt{2^{k-i}})$ by Lemma 8, whereas the second is $\Omega(2^{k-j})$ by construction, yielding the desired bound, which we show in the full version [2, Lemma 15]. ◀

It follows from Lemma 10 that if a clique in G_r has vertices in levels i and $j \geq i$ with $\Delta = j - i$, then all further vertices of the clique in larger levels are in levels $j, \dots, j + \Delta + 6$. Applying this to all pairs of vertices in a clique, we obtain that the distances $|i - j|$ between the levels i, j shrink exponentially. We formalize this in the following lemma.

▶ **Lemma 11.** *Let $r \geq 1$, $c \geq 0$ be integers, and let $S \subseteq [r]$ be a set such that for every three elements $i < j < k$ of S, we have $k - j \leq j - i + c$. Then $|S| < \log_2(r) + c + 2$.*

Proof sketch. We claim that for an optimal set S^*, we may choose the elements greedily, adding always the largest element so that $k - j \leq j - i + c$ is maintained for every $i < j < k$ in S^*. In [2, Lemma 16] we show that S^* contains roughly the c largest elements and starting from there, leaves exponentially growing gaps when smaller elements are added. ◀

Putting everything together, we conclude that the clique number of G_r is indeed $O(\log \log n)$.

▶ **Lemma 12.** *The clique number of G_r is at most $O(\log r) = O(\log \log n)$, where n is the number of vertices of G_r.*

Proof. Consider a clique C in G_r whose size we aim to bound. First recall that each level induces a cycle and thus has clique number 2. Hence, up to a factor of 2, we may assume that the vertices of C are in pairwise distinct levels. Now each three vertices of C in levels $i < j < k$ form a triangle and thus satisfy $k - j \leq j - i + 6$ by Lemma 10. Then, Lemma 11 shows that C contains only $2(\log_2(r) + 8) \in O(\log r) = O(\log \log n)$ vertices. ◀

Finally, Corollary 9 and Lemma 12 together prove Theorem 6.

3.2 Proof of Theorem 7

Our goal is to construct SHUDGs with radius $r \in \Theta(\log n)$, clique number $O(1)$, and treewidth $\Omega(\log \log n)$, which implies Theorem 1. We do so by taking a subgraph of the graph G_r for $r \geq 8$ constructed in Section 3.1 that has constant clique number but unbounded treewidth. For this, let G'_r be the subgraph of G_r that is induced by the root and the vertices in levels 2^k for $k = 3, \ldots, \log_2(r)$. Recall that G_r has r levels, where the k-th level contains 2^k vertices for $k \geq 2$, and thus the r-th level contains more than half of the vertices of G_r. Thus, G'_r has $\Theta(\log r) = \Theta(\log \log n)$ levels and $\Theta(n)$ vertices, where n is the number of vertices of G_r. Further recall that in G_r, and thus also in G'_r, Lemma 8 shows that each two levels are connected by edges. Hence, we again obtain a clique-minor by contracting each level.

▶ **Lemma 13.** *The treewidth of G'_r is at least $\Omega(\log r) = \Omega(\log \log n)$, where n is the number of vertices of G'_r.*

To bound the clique number, recall that the distance between two levels i and j refers to $|i - j|$, and not, e.g., to the hyperbolic distance.

▶ **Lemma 14.** *The clique number of G'_r is at most 5.*

Proof sketch. For the main idea, recall from Lemmas 10 and 11 that in G_r, the distances between the levels used by a clique shrink exponentially from the root to the outermost level, i.e., most vertices of any clique are in the outer levels. In contrast, we choose the levels for G'_r with exponentially growing distances, i.e., mostly inner levels are chosen. Thus, for every clique in G_r, the levels chosen for G'_r hit only very few vertices of the clique. We formalize this in the full version [2, Lemma 19]. ◀

This concludes the proof that hyperbolic uniform disk graphs do not admit product structure as Lemmas 13 and 14 together show Theorem 7, which in turn proves Theorem 1.

3.3 Dependence on the Disk Radius

The graph families we construct for Theorems 6 and 7 have disk radius $\Theta(\log n)$, which we extend to arbitrary radii by adding more vertices or stopping the construction earlier, see the full version [2]. Note that for super-constant r, we obtain unbounded row-treewidth thus no product structure. This proves the second part of Theorem 2 and strengthens Theorem 4.

▶ **Corollary 15.** *For every $r \in O(\log n)$, there are families of n-vertex HUDGs with disk radius in $\Theta(r)$,*
- *clique number in $O(\log r)$, and (row-)treewidth in $\Omega(r)$, respectively*
- *clique number in $O(1)$, and (row-)treewidth in $\Omega(\log r)$.*

▶ **Corollary 16.** *For every $r \in \Omega(\log n)$, there are families of n-vertex SHUDGs with disk radius in $\Theta(r)$,*
- *clique number in $O(\log \log n)$, and (row-)treewidth in $\Omega(\log n)$, respectively*
- *clique number in $O(1)$, and (row-)treewidth in $\Omega(\log \log n)$.*

4 Product Structure for Small Disk Radius

In this section, we show that HUDGs whose disk radius and clique number are (sub-)constant admit product structure, which finishes the proof of Theorem 2. In fact, we show that every HUDG G with disk radius r and clique number ω is a subgraph of $H \boxtimes K_k$, where H is a hyperbolic tiling and $k \in O(\omega \cdot 3^{8r})$, which emphasizes their grid-structure. Since H is planar, and planar graphs are subgraphs of $H' \boxtimes P \boxtimes K_3$ with $\text{tw}(H') = 3$ [16], it follows that $G \subseteq H' \boxtimes P \boxtimes K_{3k}$. This implies that $G \subseteq H'' \boxtimes P$ for the graph $H'' = H' \boxtimes K_{3k}$ of treewidth $O(\omega \cdot 3^{8r})$, i.e., G has row-treewidth $O(\omega \cdot 3^{8r})$. Thus, in the following theorem, the product $H \boxtimes K_{O(\omega \cdot 3^{8r})}$ is the strongest statement and the others follow.

▶ **Theorem 17.** *For every HUDG G with disk radius r and clique size ω it holds that $G \subseteq H \boxtimes K_{O(\omega \cdot 3^{8r})} \subseteq H' \boxtimes P \boxtimes K_{O(\omega \cdot 3^{8r})}$, where H is a (possibly irregular) hyperbolic tiling and H' is a graph of treewidth at most 3. Moreover, the row-treewidth of G is in $O(\omega \cdot 3^{8r})$.*

Note that Theorem 17 indeed implies the first part of Theorem 2 since the row-treewidth only depends on ω and r, and thus we obtain product structure for every graph class with (sub-)constant clique number and disk radius. Further note that we also obtain an upper bound on the row-treewidth if clique number or disk radius are super-constant for some family of HUDGs. However, in these cases the upper bound on the row-treewidth also grows and thus we do not obtain product structure, which is no surprise given Section 3.

Recall that [20] shows that EUDGs with clique number ω are subgraphs of $H \boxtimes K_{O(\omega)}$, where H is a square-grid with diagonals. To find the product $H \boxtimes K_k$ for HUDGs, we lift their approach to the hyperbolic setting with two major changes. Not surprisingly, we replace the Euclidean tiling by a hyperbolic tiling to adjust to the geometry. However, we also need to face different disk radii since, in contrast to [20], we cannot freely choose the size of our tiles. For large disk radii, we solve this by merging tiles, whereas we subdivide tiles for small disk radii as needed. To do so, we provide a geometric interpretation of what we aim for. Two tiles are called *adjacent* if they are adjacent in the dual, i.e., if they share an edge.

▶ **Theorem 3.** *For each $r > 0$ there is an (irregular) tiling of the hyperbolic plane such that*
- *every tile can be covered by $O(3^{8r})$ disks of radius r and*
- *each two points with distance at most $2r$ lie in the same tile or in two adjacent tiles.*

Observe that this indeed implies that every HUDG with radius r and clique number ω is a subgraph of $H \boxtimes K_k$, where H is the dual of the tiling from Theorem 3, and $k \in O(\omega \cdot 3^{8r})$. To see this, consider the partition $\mathcal{P}$ with one part for the vertices of each tile. By the first property, each part contains at most $O(\omega \cdot 3^{8r})$ vertices since each disk of radius r covers a clique and thus at most ω vertices. Thus, we have $G \subseteq Q \boxtimes K_k$ for the quotient $Q = G/\mathcal{P}$ that is obtained by contracting the vertices in each tile, and some $k \in O(\omega \cdot 3^{8r})$. Then, by

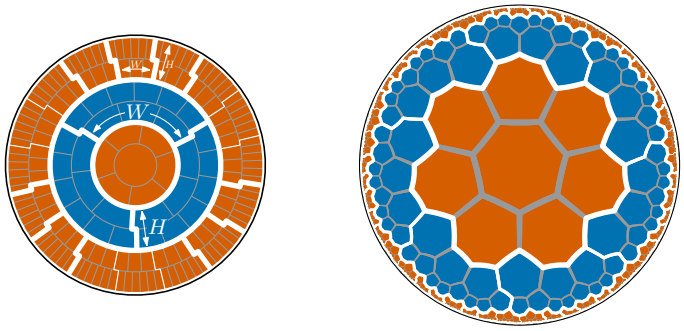


Figure 8 To ensure that non-adjacent tiles are sufficiently far from each other, we merge 7-gons as indicated by the white lines. High-level, we define rings of height $H \approx 2r$, which we cut into parts of width $W \in O(3^{4r})$ (left). Locally, however, we stick to the 7-gons from a $\{7, 3\}$-tiling (right).

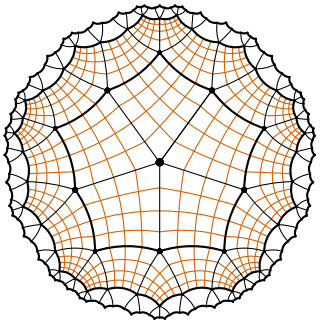


Figure 9 A $\{4, 5\}$-tiling (black) that is subdivided (red) so that each small tile can be covered by $O(1)$ disks of radius r.

the second property, the endpoints of every edge lie in the same or in adjacent tiles, so the quotient Q is a subgraph of the dual H of the tiling from Theorem 3. Together, we obtain $G \subseteq Q \boxtimes K_k \subseteq H \boxtimes K_k$, as required by Theorem 17.

The main challenge of this section is to prove Theorem 3, for which we refer to the full version [2]. For an overview, if r is at least some carefully chosen constant, we start with a regular $\{7, 3\}$-tiling whose tiles are merged until the second property of Theorem 3 is satisfied (Figure 8). Here, the tiles have a rapidly growing maximum degree to meet the requirements of the hyperbolic plane. The case for small radii then starts with a regular $\{4, 5\}$-tiling, where we subdivide the tiles until they are small enough for the first property (see Figure 9). This results in a tiling that gets more Euclidean the smaller r is, in the sense that locally, it looks like a Euclidean square grid.

5 Open Questions

In this paper, we characterize for which functions r families of HUDGs with constant clique number and disk radius r admit product structure: If $r \in O(1)$, then every family has bounded row-treewidth and thus admits product structure, otherwise there are families with unbounded row-treewidth. However, our upper bound is exponential in the disk radius, whereas our lower bound is logarithmic. It is an interesting open question whether the exponential dependency is necessary, both for Theorems 3 and 17. We conjecture that it is indeed necessary for the tiling, but believe that the base can be improved, which would also improve the bound on the row-treewidth. Note that even if there is an exponential lower bound for the tiling, it does not necessarily transfer to the row-treewidth of HUDGs.

▶ **Question 18.** *What is the row-treewidth of HUDGs with clique number ω and disk radius r?*

With our lower-bound construction, we show that there are SHUDGs with radius $\Theta(\log n)$ having treewidth $\Omega(\omega \frac{\log n}{\log \log n})$. This almost matches the upper bound of $O(\omega \cdot \log n)$ [5, 21]. We ask whether the remaining gap can be closed. Moreover, we ask whether similar lower bounds are possible for other clique numbers than $\omega \in O(\log \log n)$; specifically for constant ω.

▶ **Question 19.** *Is there a family of n-vertex HUDGs with clique number ω and treewidth $\Theta(\omega \cdot \log n)$? Is there a family with constant clique number and treewidth $\Theta(\log n)$?*

Finally, we conjecture that our lower bound for the row-treewidth stops growing at logarithmic grid radius because there are no HUDGs that require larger radius.

▶ **Question 20.** *Are there families of HUDGs that do not admit a disk representation in the hyperbolic plane such that the disk radius is $O(\log n)$?*

References

1 Shinwoo An, Seonghyuk Im, Seokbeom Kim, and Myounghwan Lee. An efficient algorithm for $\mathcal{F}$-subgraph-free edge deletion on graphs having a product structure, 2025. doi:10.48550/arXiv.2510.14674.

2 Thomas Bläsius, Emil Dohse, Deborah Haun, and Laura Merker. Product structure and treewidth of hyperbolic uniform disk graphs, 2026. doi:10.48550/arXiv.2603.18997.

3 Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, and Daniel Stephan. Strongly hyperbolic unit disk graphs. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 13:1–13:17, 2023. doi:10.4230/LIPIcs.STACS.2023.13.

4 Thomas Bläsius, Tobias Friedrich, and Anton Krohmer. Hyperbolic random graphs: Separators and treewidth. In *European Symposium on Algorithms (ESA)*, pages 15:1–15:16, 2016. doi:10.4230/LIPIcs.ESA.2016.15.

5 Thomas Bläsius, Jean-Pierre von der Heydt, Sándor Kisfaludi-Bak, Marcus Wilhelm, and Geert van Wordragen. Structure and independence in hyperbolic uniform disk graphs. In *Symposium on Computational Geometry (SoCG)*, pages 21:1–21:16, 2025. doi:10.4230/LIPIcs.SoCG.2025.21.

6 Prosenjit Bose, Pat Morin, and Saeed Odak. An optimal algorithm for product structure in planar graphs. In *Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 19:1–19:14, 2022. doi:10.4230/LIPIcs.SWAT.2022.19.

7 Rutger Campbell, Katie Clinch, Marc Distel, J. Pascal Gollin, Kevin Hendrey, Robert Hickingbotham, Tony Huynh, Freddie Illingworth, Youri Tamitegama, Jane Tan, and et al. Product structure of graph classes with bounded treewidth. *Combinatorics, Probability and Computing*, 33(3):351–376, 2024. doi:10.1017/S0963548323000457.

8 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165–177, 1990. doi:10.1016/0012-365X(90)90358-O.

9 H. S. M. Coxeter. The trigonometry of hyperbolic tessellations. *Canadian Mathematical Bulletin*, 40(2):158–168, 1997. doi:10.4153/CMB-1997-019-0.

10 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015. doi:10.1007/978-3-319-21275-3.

11 Clément Dallard, Martin Milanič, Andrea Munaro, and Shizhou Yang. Layered tree-independence number and clique-based separators, 2025. doi:10.48550/arXiv.2506.12424.

12 Clément Dallard, Martin Milanič, and Kenny Štorgel. Treewidth versus clique number. II. tree-independence number. *Journal of Combinatorial Theory, Series B*, 164:404–442, 2024. doi:10.1016/j.jctb.2023.10.006.

13 Basudeb Datta and Subhojoy Gupta. Semi-regular tilings of the hyperbolic plane. *Discrete Comput Geom*, pages 531–553, 2019. doi:10.1007/s00454-019-00156-0.

14 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for exponential-time-hypothesis–tight algorithms and lower bounds in geometric intersection graphs. *SIAM Journal on Computing*, 49(6):1291–1331, 2020. doi:10.1137/20M1320870.

15 Marc Distel, Robert Hickingbotham, Tony Huynh, and David R. Wood. Improved product structure for graphs on surfaces. *Discrete Mathematics & Theoretical Computer Science*, 24, 2022. doi:10.46298/dmtcs.8877.

16 Vida Dujmović, Gwenaël Joret, Piotr Micek, Pat Morin, Torsten Ueckerdt, and David R. Wood. Planar graphs have bounded queue-number. *J. ACM*, 67(4), 2020. doi:10.1145/3385731.

17 Vida Dujmović, Gwenaël Joret, Piotr Micek, Pat Morin, Torsten Ueckerdt, and David Wood. Planar graphs have bounded queue-number. In *Symposium on Foundations of Computer Science (FOCS)*, pages 862–875, 2019. doi:10.1109/FOCS.2019.00056.

18 Vida Dujmović, Pat Morin, and David R. Wood. Layered separators in minor-closed graph classes with applications. *Journal of Combinatorial Theory, Series B*, 127:111–147, 2017. doi:10.1016/j.jctb.2017.05.006.

19 Vida Dujmović, Pat Morin, and David R. Wood. Graph product structure for non-minor-closed classes. *Journal of Combinatorial Theory, Series B*, 162:34–67, 2023. doi:10.1016/j.jctb.2023.03.004.

20 Zdeněk Dvořák, Tony Huynh, Gwenael Joret, Chun-Hung Liu, and David R. Wood. *Notes on Graph Product Structure Theory*, pages 513–533. Springer International Publishing, 2021. doi:10.1007/978-3-030-62497-2_32.

21 Zdeněk Dvořák and Sergey Norin. Treewidth of graphs with balanced separations. *Journal of Combinatorial Theory, Series B*, 137:137–144, 2019. doi:10.1016/j.jctb.2018.12.007.

22 Tobias Friedrich and Anton Krohmer. On the diameter of hyperbolic random graphs. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 614–625. Springer-Verlag, 2015. doi:10.1007/978-3-662-47666-6_49.

23 Tobias Friedrich and Anton Krohmer. On the diameter of hyperbolic random graphs. *SIAM Journal on Discrete Mathematics*, 32(2):1314–1334, 2018. doi:10.1137/17M1123961.

24 Esther Galby, Andrea Munaro, and Shizhou Yang. Polynomial-time approximation schemes for independent packing problems on fractionally tree-independence-number-fragile graphs. In *Symposium on Computational Geometry (SoCG)*, volume 258, pages 34:1–34:15, 2023. doi:10.4230/LIPIcs.SoCG.2023.34.

25 Marvin Jay Greenberg. *Euclidean and Non-Euclidean Geometries*. W. H. Freeman and Company, 3rd edition, 1993.

26 Luca Gugelmann, Konstantinos Panagiotou, and Ueli Peter. Random hyperbolic graphs: Degree sequence and clustering. In *Automata, Languages, and Programming (ICALP)*, pages 573–585, 2012. doi:10.1007/978-3-642-31585-5_51.

27 Robert Hickingbotham, Paul Jungeblut, Laura Merker, and David R. Wood. The product structure of squaregraphs. *Journal of Graph Theory*, 105(2):179–191, 2024. doi:10.1002/jgt.23008.

28 Robert Hickingbotham and David R. Wood. Shallow minors, graph products, and beyond-planar graphs. *SIAM Journal on Discrete Mathematics*, 38(1):1057–1089, 2024. doi:10.1137/22M1540296.

29 Nikolai Karol. String graphs: Product structure and localised representations, 2025. doi:10.48550/arXiv.2511.15156.

30 Sándor Kisfaludi-Bak. Hyperbolic intersection graphs and (quasi)-polynomial time. In *Symposium on Discrete Algorithms (SODA)*, pages 1621–1638, 2020. doi:10.1137/1.9781611975994.100.

31 Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010. doi:10.1103/PhysRevE.82.036106.

32 Laura Merker, Lena Scherzer, Samuel Schneider, and Torsten Ueckerdt. Intersection graphs with and without product structure. In *International Symposium on Graph Drawing and Network Visualization (GD)*, volume 320, pages 23:1–23:19, 2024. doi:10.4230/LIPIcs.GD.2024.23.

33 Yanick Thurn, Manuel Schrauth, and Johanna Erdmenger. Hyperbolic tiling neighborhoods in O(1) time, 2025. doi:10.48550/arXiv.2508.04765.

Splitting Sandwiches Unevenly via Unique Sink Orientations and Rainbow Arrangements

Michaela Borzechowski
Department of Mathematics and Computer Science, Freie Universität Berlin, Germany

Sebastian Haslebacher
Department of Computer Science, ETH Zürich, Switzerland

Hung P. Hoang
Algorithms and Complexity Group, Faculty of Informatics, TU Wien, Austria

Patrick Schnider
Department of Mathematics and Computer Science, University of Basel, Switzerland
Department of Computer Science, ETH Zürich, Switzerland

Simon Weber
Department of Computer Science, ETH Zürich, Switzerland

Abstract

The famous Ham-Sandwich theorem states that any d point sets in $\mathbb{R}^d$ can be simultaneously bisected by a single hyperplane. The α-Ham-Sandwich theorem gives a sufficient condition for the existence of biased cuts, i.e., hyperplanes that do not cut off half but some prescribed fraction of each point set. We give two new proofs for this theorem. The first proof is completely combinatorial and highlights a strong connection between the α-Ham-Sandwich theorem and Unique Sink Orientations of grids. The second proof uses point-hyperplane duality and the Poincaré-Miranda theorem and allows us to generalize the result to and beyond oriented matroids. For this we introduce a new concept of rainbow arrangements, generalizing colored pseudo-hyperplane arrangements. Along the way, we also show that the realizability problem for rainbow arrangements is $\exists\mathbb{R}$-complete, which also implies that the realizability problem for grid Unique Sink Orientations is $\exists\mathbb{R}$-complete.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Problems, reductions and completeness; Mathematics of computing → Discrete mathematics

Keywords and phrases α-Ham-Sandwich Theorem, Pseudo-Hyperplanes, Arrangements, Unique Sink Orientations, Oriented Matroids

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.19

Related Version *Full Version*: `https://arxiv.org/abs/2602.10795` [11]

Funding *Michaela Borzechowski*: DFG within GRK 2434 *Facets of Complexity*.
Hung P. Hoang: Austrian Science Foundation (FWF, projects 10.55776/Y1329 and ESP1136425).

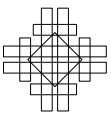
1 Introduction

The famous *Ham-Sandwich theorem*, originally proven by Stone and Tukey in 1942 [30], states that given any d mass partitions or point sets in $\mathbb{R}^d$, there is a hyperplane that simultaneously bisects all of them. As the name suggests, this can be illustrated using the following food-based analogy: assume you have a 3-dimensional sandwich consisting of bread, ham, and cheese, that you want to share with your friend. You want to do this fairly, meaning that both of you get exactly half of the bread, half of the ham, and half of the cheese. The Ham-Sandwich theorem now says that you can always get a fair division with a single straight cut, no matter how you assembled the sandwich[1].

[1] In his book "Algorithms in Combinatorial Geometry", Edelsbrunner illustrates this by saying that such a cut can be found even if the cheese is still in the fridge [12].



© Michaela Borzechowski, Sebastian Haslebacher, Hung P. Hoang, Patrick Schnider, and Simon Weber;
licensed under Creative Commons License CC-BY 4.0



The Ham-Sandwich theorem has initiated the study of *mass partitions*, where the general question is how many mass distributions or point sets can be simultaneously bisected with some specific type of cut. Studied variants include partitions with several hyperplanes [7, 9, 18, 19, 25], partitions using general convex sets [1, 3, 8], partitions using fans and cones [6, 24, 27], or partitions using fixed shapes [26]. For more information on mass partitions, we refer to the recent survey by Roldán-Pensado and Soberón [23].

If you like cheese significantly more than your friend does, you might not want to share your sandwich fairly, but you perhaps want to have more of the cheese. A natural question is which such biased partitions are still possible with just a single cut. The *α-Ham-Sandwich theorem* gives a sufficient condition for when such cuts are possible. In particular, it states that if your ingredients are *well-separated*, then any biased cut is possible. Formally, well-separation is defined as follows (see also Figure 1 for an illustration).

▶ **Definition 1.** *A family $\mathcal{F} = (A_1, \dots, A_d)$ of d point sets (or mass distributions) is said to be* well-separated *if for any index set $I \subset [d]$,[2] there exists a hyperplane that separates the convex hull of (the support of) $\bigcup_{i\in I} A_i$ from the convex hull of (the support of) $\bigcup_{i\in[d]\setminus I} A_i$.*

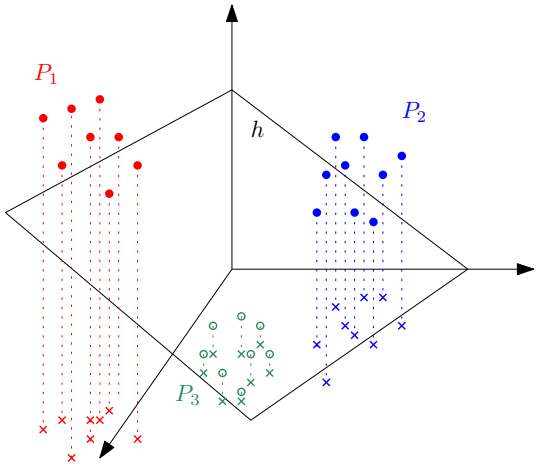


Figure 1 A family of well-separated point sets $\mathcal{F} = (P_1, P_2, P_3)$ in $\mathbb{R}^3$. The hyperplane h separates $P_1 \cup P_2$ from P_3.

The α-Ham-Sandwich theorem has been shown both for mass distributions, by Bárány, Hubard, and Jerónimo [5], as well as for finite point sets, by Steiger and Zhao [28]. In this manuscript, we focus on the discrete variant for point sets, so we only formally state this version here. However, before getting there, let us remark that we orient a hyperplane $h \in \mathbb{R}^d$ defined by points $p_1 \in P_1, \dots, p_d \in P_d$ according to the orientation of the simplex $(p_1, \dots, p_d) \in h$, that is, a point q is below h iff the following $(d+1) \times (d+1)$ matrix A has a negative determinant: $A_{j,d+1} = 1$ for all $j \in [d+1]$, $A_{i,\cdot} = p_i$ for all $i \in [d]$ and $A_{d+1,\cdot} = q$. If the matrix A has a positive determinant we say that q is above h. If q is on h, then A has determinant 0.

With this, we can now formally define α-cuts (see also Figure 2) and state the discrete α-Ham-Sandwich theorem.

[2] Herein and henceforth, $[d]$ refers to the set $\{1, \dots, d\}$.

▶ **Definition 2.** *Let $\mathcal{P} = (P_1 \dots, P_d)$ be well-separated and in weak general position*[3]*. Given $(\alpha_1, \dots, \alpha_d) \in [|P_1|] \times \cdots \times [|P_d|]$, an* $(\alpha_1, \dots, \alpha_d)$-cut *is a hyperplane h_α such that for each $i \in [d]$, one point of P_i is on h_α and exactly $\alpha_i - 1$ points of P_i are below it.*

▶ **Theorem 3.** *Let $\mathcal{P} = (P_1 \dots, P_d) \subset \mathbb{R}^d$ be well-separated and in weak general position. Then for every $(\alpha_1, \dots, \alpha_d) \in [|P_1|] \times \cdots \times [|P_d|]$, there exists a unique $(\alpha_1, \dots, \alpha_d)$-cut.*

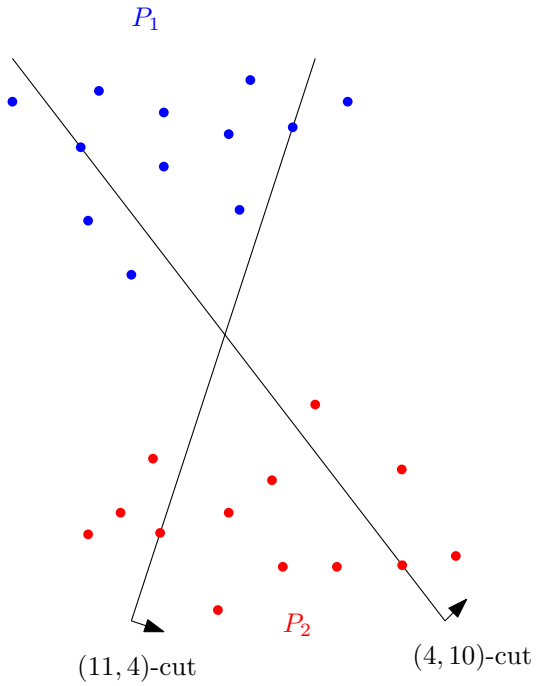


Figure 2 Well-separated point sets in $\mathbb{R}^2$ with two α-cuts.

The proof of the continuous version for mass distributions by Bárány, Hubard, and Jerónimo [5] uses Brouwer's fixpoint theorem. On the other hand, the proof of the discrete version for point sets by Steiger and Zhao [28] uses an elegant inductive argument to show that every α-cut is unique and then deduces the existence from the pigeonhole principle. However, we think that there is a gap in their inductive argument, which we elaborate on in the appendix of the full version of this paper. While it is possible that their proof can be fixed using some additional arguments, we are taking a different route and instead provide two new proofs of the α-Ham-Sandwich theorem for point sets.

The first proof, presented in Section 2, uses the combinatorial framework of *Unique Sink Orientations*. A Unique Sink Orientation (USO) is an orientation of the edge set of a hypercube such that every subcube has a unique sink. USOs were formally defined by Szabó and Welzl [31], based on earlier work by Stickney and Watson [29] who investigated USOs as a combinatorial abstraction of the candidate solutions of a special class of linear complementarity problems. The concept of USOs has been generalized to *grids*, which are products of complete graphs [15]. A *grid USO* is an orientation of a grid where every subgrid has a unique sink.

[3] Weak general position means that any hyperplane containing one point of each of the sets $P_1, \dots, P_d$ cannot contain any other points of $\mathcal{P}$.

Our second new proof of the α-Ham-Sandwich theorem, presented in Section 3, considers the setting under the well-known point-hyperplane duality: given d sets of hyperplanes with a dual notion of "well-separated", we prove the existence of a point that lies above or below the correct number of hyperplanes in each set. In fact, we present a more general result, replacing the arrangement of hyperplanes by a *rainbow arrangement*, a concept that we formally introduce in Section 3.1. Informally, a colored generalized arrangement is a family $\mathcal{F} = (H_1, \dots, H_d)$, where each H_i is a set of pseudo-hyperplanes in $\mathbb{R}^d$. In a rainbow arrangement, we further require that any colorful choice of pseudo-hyperplanes $h_1 \in H_1, \dots, h_d \in H_d$ intersect in a single point. The main result of Section 3.3 is the following theorem.

▶ **Theorem 4.** *Let $\mathcal{F} = (H_1, \dots, H_d) \subset \mathbb{R}^d$ be a well-separated colored generalized arrangement where each color class H_i has size n_i. Then for every $(\alpha_1, \dots, \alpha_d) \in [n_1] \times \cdots \times [n_d]$, there is a point $x_\alpha \in \mathbb{R}^d$ lying on one and above exactly $\alpha_i - 1$ pseudo-hyperplanes of color i for all $i \in [d]$. If $\mathcal{F}$ is additionally a rainbow arrangement, then the point x_α is unique.*

Using the topological representation theorem, Theorem 4 allows us to deduce a version of the α-Ham-Sandwich theorem for oriented matroids (Corollary 11).

In Section 4, we generalize Theorem 4 even further by relaxing the well-separation condition to the notion of (β, γ)-*separation*, which we introduce in the same section.

Finally, our notion of rainbow arrangements can be viewed as a higher-dimensional generalization of *bicolored order types*, introduced by Aichholzer and Brötzner [2]. In Section 5, we show that deciding whether a rainbow arrangement can be realized by a hyperplane arrangement is $\exists\mathbb{R}$-complete already in two dimensions (i.e., for bicolored order types, see Theorem 21). Given the connection between the α-Ham-Sandwich theorem and grid USOs, and based on recent results by Borzechowski, Fearnley, Gordon, Savani, Schnider, and Weber [10], this allows us to conclude that deciding whether a grid USO is realizable is also $\exists\mathbb{R}$-complete (see Corollary 22).

2 First Proof: Unique Sink Orientations

A grid graph is a generalization of the d-dimensional hypercube: While the latter is the Cartesian product of d copies of K_2 (the complete graph on two vertices), the former is the Cartesian product of complete graphs of arbitrary size. Formally, a d-dimensional *grid graph* Γ parameterized by $n_1, \dots, n_d \in \mathbb{N}_{\geq 2}$ is the graph on $[n_1] \times \cdots \times [n_d]$, where two vertices are adjacent if and only if they differ in exactly one coordinate. We say the grid has d *dimensions* and each dimension i has n_i *directions*.

The subgraph Γ' of Γ induced by the vertices $V(\Gamma') = N_1 \times \cdots \times N_d$ for non-empty $N_i \subseteq [n_i]$ is called an *induced subgrid* of Γ. Note that if we have $|N_i| = 1$ for some i, then the induced subgrid loses a dimension. If $|N_i| \leq 2$ for all $i \in [d]$, we say that Γ' is a *subcube* of Γ. An orientation of the edges of a grid graph is called a *Unique Sink Orientation (USO)* if every induced subgrid has a unique sink (i.e., a unique vertex that has no outgoing edges) [15].

▶ **Lemma 5.** *Let σ be a grid orientation. Assume that every induced subgrid has a (not necessarily unique) sink, and that σ is a USO when restricted to any subcube. Then σ is a USO.*

Proof. As every subgrid has a sink by assumption, it remains to show that this sink is unique. Assume for the sake of contradiction that there is a subgrid that has two sinks $a = (a_1, \dots, a_d)$ and $b = (b_1, \dots, b_d)$. Consider now the subcube spanned by a and b (i.e.,

the subcube induced by $\{a_1, b_1\} \times \cdots \times \{a_d, b_d\}$). As a and b were sinks in the subgrid, they are also sinks in this subcube. We have thus found a subcube with two sinks, so σ restricted to this subcube is not a USO. This is a contradiction to the second assumption. ◀

Let $\mathcal{P} = (P_1, \dots, P_d) \subset \mathbb{R}^d$ be a well-separated point set in weak general position. For $i \in [d]$, let $P_i = \{p_1^i, \dots, p_{|P_i|}^i\}$. Let $\Gamma_\mathcal{P}$ be the grid graph parameterized by $|P_1|, \dots, |P_d|$. We recall the following orientation $\sigma_\mathcal{P}$ on $\Gamma_\mathcal{P}$, which was presented in [10] and is illustrated in Figure 3. For any edge between two vertices $v = (a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_d)$ and $v' = (a_1, \dots, a_{i-1}, a'_i, a_{i+1}, \dots, a_d)$ of $\Gamma_\mathcal{P}$, let h be the colorful hyperplane spanned by the points $p^1_{a_1}, \dots, p^{i-1}_{a_{i-1}}, p^i_{a_i}, p^{i+1}_{a_{i+1}}, \dots, p^d_d$, and let h' be the colorful hyperplane spanned by $p^1_{a_1}, \dots, p^{i-1}_{a_{i-1}}, p^i_{a'_i}, p^{i+1}_{a_{i+1}}, \dots, p^d_d$. If $p^i_{a'_i}$ lies above h, we orient v' towards v; otherwise orient v towards v'. It turns out that $p^i_{a'_i}$ lies above h if and only if $p^i_{a_i}$ lies below the hyperplane h',[4] so this orientation is well-defined.

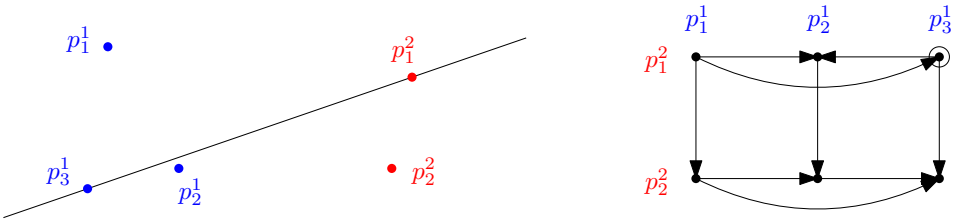


Figure 3 Well-separated point sets in $\mathbb{R}^2$ with the corresponding grid orientation. The line drawn on the left is represented by the highlighted vertex on the right.

The following lemma is key to our first proof of Theorem 3.

▶ **Lemma 6.** *Let $\mathcal{P} = (P_1, \dots, P_d) \subset \mathbb{R}^d$ be well-separated and in weak general position. Then the orientation $\sigma_\mathcal{P}$ is a grid USO.*

Note that this lemma was already proved in [10]. However, their proof uses Theorem 3. To avoid the cyclic dependency, we provide here an alternative proof that uses the following version of the α-Ham-Sandwich theorem for mass distributions.

▶ **Theorem 7** (Bárány, Hubard, and Jerónimo [5])**.** *Let $K_1, \dots, K_d$ be well-separated convex bodies in $\mathbb{R}^d$, and $\alpha_1, \dots, \alpha_d$ given constants with $0 \le \alpha_i \le 1$ for $i \in [d]$. Then there is a unique hyperplane h such that for all $i \in [d]$, h intersects K_i, and the proportion of the volume of K_i below h is α_i.*

Proof of Lemma 6. We first show that every subgrid has a sink. By the definition of $\sigma_\mathcal{P}$, a subgrid is spanned by subsets $P'_i \subseteq P_i$ and a sink corresponds to a $(1, \dots, 1)$-cut. As any subset of a well-separated point set in weak general position is again well-separated and in weak general position, the existence of such a $(1, \dots, 1)$-cut is guaranteed by applying Theorem 7 with the convex sets being the convex hulls of $P'_1, \dots, P'_d$, and $\alpha_i = 0$ for $i \in [d]$.

We now argue that every subcube is a USO. A subcube is spanned by at most two points per P_i. Let Q_i be the set of these at most two points. As argued before, $(Q_1, \dots, Q_d)$ is well-separated and in weak general position. Further, the convex hull of the at most two points per P_i is a line segment. Hence, by applying Theorem 7 on these convex hulls, we obtain the existence of all possible $(\alpha_1, \dots, \alpha_d)$-cuts of $(Q_1, \dots, Q_d)$ for all $(\alpha_1, \dots, \alpha_d) \in$

[4] This can be seen by projecting to $\mathbb{R}^2$ such that $h \cap h'$ is mapped to the origin.

$[|Q_1|] \times \cdots \times [|Q_d|]$. Hence, in the subcube, each possible *outmap* (i.e., a binary vector at each vertex that encodes whether the incident edge along each dimension is incoming or outgoing) is present. This implies that the subcube is indeed a USO [31].

The claim now follows from Lemma 5. ◀

The α-Ham-Sandwich theorem for point sets (Theorem 3) now follows easily.

▶ **Theorem 3.** *Let $\mathcal{P} = (P_1 \dots, P_d) \subset \mathbb{R}^d$ be well-separated and in weak general position. Then for every $(\alpha_1, \dots, \alpha_d) \in [|P_1|] \times \cdots \times [|P_d|]$, there exists a unique $(\alpha_1, \dots, \alpha_d)$-cut.*

Proof. By Lemma 6, the orientation $\sigma_{\mathcal{P}}$ is a grid USO. We define a function $r : [|P_1|] \times \cdots \times [|P_d|] \to \{0, \dots, |P_1| - 1\} \times \cdots \times \{0, \dots, |P_d| - 1\}$ that assigns to each vertex of the grid graph $\Gamma_{\mathcal{P}}$ a tuple $(a_1, \dots, a_d)$ such that a_i is the number of the vertex's outgoing edges in the dimension i. Then [15, Theorem 2.14] states that r is a bijection. This implies that for every $(\alpha_1, \dots, \alpha_d) \in [|P_1|] \times \cdots \times [|P_d|]$, there is a unique $(\alpha_1, \dots, \alpha_d)$-cut, as required. ◀

3 Second Proof: Rainbow Arrangements and the Poincaré-Miranda Theorem

We provide definitions and preliminaries on the Poincaré-Miranda Theorem in Sections 3.1 and 3.2 and present the proof of Theorem 4 in Section 3.3.

3.1 Point-Hyperplane Duality and Rainbow Arrangements

In this subsection, we argue why our α-Ham-Sandwich theorem for rainbow arrangements (Theorem 4) is a generalization of the one for point sets (Theorem 3). We first consider the point-hyperplane dual of Theorem 3. The dual of every input point is a hyperplane, and the well-separation of the point set translates to the property that for every subset S of $[d]$, there exists a point above all the dual hyperplanes of the points in $\cup_{i \in S} P_i$ and below those of the points in $\cup_{i \notin S} P_i$. The dual of an $(\alpha_1, \dots, \alpha_d)$-cut is then a point lying on one dual hyperplane and above exactly $\alpha_i - 1$ hyperplanes of each color i. Theorem 3 is equivalent to the statement that such a point is unique for every $(\alpha_1, \dots, \alpha_d)$.

With this interpretation in mind, we now define our generalized arrangements, where we define an *oriented pseudo-hyperplane* as a subset of $\mathbb{R}^d$ that is homeomorphic to $\mathbb{R}^{d-1}$ and divides $\mathbb{R}^d$ into two parts: a positive and a negative side. We say a point p lies *above* (*below*) an oriented pseudo-hyperplane H if and only if p is contained in the positive (negative) side.

▶ **Definition 8** (Generalized Arrangement, Rainbow Arrangement). *A family $\mathcal{F}$ of oriented pseudo-hyperplanes in $\mathbb{R}^d$ is called a* generalized arrangement *if for all $k \in [d+1]$, the intersection of any k of them is a (not necessarily connected) $(d-k)$-dimensional manifold*[5] *and $\mathbb{R}^d \setminus \bigcup \mathcal{F}$ has finitely many connected components (which we call* cells*).*

A generalized arrangement is colored *if $\mathcal{F}$ is partitioned into d pairwise disjoint subfamilies $H_1, \dots, H_d$. We think of this as coloring each oriented pseudo-hyperplane with one of d colors and call the H_i's* color classes*.*

A colored generalized arrangement is a rainbow arrangement *if for each choice of one oriented pseudo-hyperplane per color class, $h_1 \in H_1, \dots, h_d \in H_d$, we have that the intersection $h_1 \cap \dots \cap h_d$ is a single point.*

[5] For the sake of convenience, we consider the empty set to be a manifold of any dimension. In particular, it is the only manifold of dimension -1.

Note that our arrangements differ significantly from the well-known concept of *pseudo-hyperplane arrangements*, where the intersection of any k pseudo-hyperplanes is required to be homeomorphic to $\mathbb{R}^{d-k}$. In particular, while every pseudo-hyperplane arrangement is also a generalized arrangement, the converse is not true.

The following definition of well-separation is a dual version of the notion of well-separation for point sets.

▶ **Definition 9** (Well-Separated). *A colored generalized arrangement $\mathcal{F} = (H_1, \dots, H_d)$ in $\mathbb{R}^d$ is* well-separated *if for every sign vector $s \in \{+,-\}^d$, there exists a point $x_s \in \mathbb{R}^d$ such that*

- *x_s lies in an unbounded cell,*
- *x_s lies above all oriented pseudo-hyperplanes in H_i iff $s_i = +$,*
- *and x_s lies below all oriented pseudo-hyperplanes in H_i iff $s_i = -$,*

for all $i \in [d]$.

3.2 The Poincaré-Miranda Theorem

In our proof, we will use the Poincaré-Miranda theorem, which we briefly recall.

▶ **Theorem 10** (Poincaré-Miranda theorem). *Consider d continuous functions*

$$f_1, \dots, f_d : [-1,1]^d \to \mathbb{R}.$$

Assume that for each variable x_i, the function f_i is non-positive whenever $x_i = -1$ and non-negative whenever $x_i = 1$. Then there exists a point x^* in $[-1,1]^d$ with $f_i(x^*) = 0$ for all $i \in \{1, \dots, d\}$.

The theorem was first stated by Poincaré [21] without proof. It was later shown by Miranda [20] that it is equivalent to Brouwer's fixed point theorem. Since then, the theorem has found numerous applications and generalizations; see, e.g., [4] and the references therein.

3.3 The Second Proof

We define the *k-level* of a family H of oriented pseudo-hyperplanes to be the set of points that lie (i) on at least one oriented pseudo-hyperplane of H and (ii) on or above exactly k oriented pseudo-hyperplanes of H. We are now ready to prove our generalization of the α-Ham-Sandwich theorem.

▶ **Theorem 4.** *Let $\mathcal{F} = (H_1, \dots, H_d) \subset \mathbb{R}^d$ be a well-separated colored generalized arrangement where each color class H_i has size n_i. Then for every $(\alpha_1, \dots, \alpha_d) \in [n_1] \times \cdots \times [n_d]$, there is a point $x_\alpha \in \mathbb{R}^d$ lying on one and above exactly $\alpha_i - 1$ pseudo-hyperplanes of color i for all $i \in [d]$. If $\mathcal{F}$ is additionally a rainbow arrangement, then the point x_α is unique.*

We prove the existence of a point x_α by showing that the corresponding α_i-levels intersect, see Figure 4 for an illustration. The uniqueness for rainbow arrangement additionally uses a counting argument.

Proof. By assumption $\mathcal{F}$ is well-separated, so for each vector $s \in \{+,-\}^d$ there is a point x_s in an unbounded cell that lies above or below all oriented pseudo-hyperplanes of a color class. Consider a subset $I \subset [d]$ and a family of vectors $s^1, \dots, s^k$ in $\{+,-\}^d$ for which $s^1_i = \dots = s^k_i$ for all $i \in I$. Let $\mathcal{F}_\mathcal{I} = \cup_{i \in I} H_i$ be the generalized arrangement consisting only of oriented pseudo-hyperplanes of color classes in I. We claim that the points $x_{s^1}, \cdots, x_{s^k}$ lie in a common unbounded cell of $\mathcal{F}_\mathcal{I}$. Indeed, assume for the sake of contradiction that

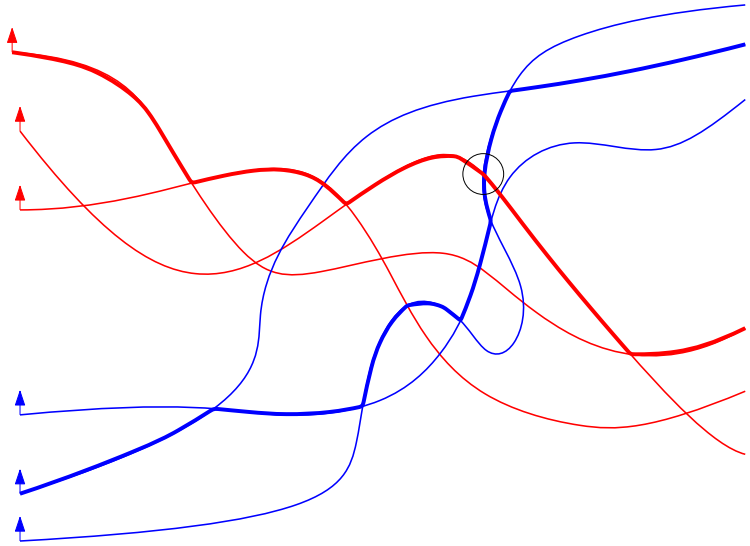


Figure 4 A rainbow arrangement with colors red and blue. The red 3-level and the blue 2-level are highlighted and intersect in a unique point.

two of the points x_{s^i} and x_{s^j} lie in different unbounded cells. Then there is an oriented pseudo-hyperplane of $\mathcal{F}_\mathcal{I}$ separating them. However, by definition, x_{s^i} and x_{s^j} must lie on the same side of any oriented pseudo-hyperplane in $\mathcal{F}_\mathcal{I}$, a contradiction.

Now consider the cube $[-1,1]^d$. Note that each face of the cube can be encoded by a vector z in $\{*,-1,1\}^d$, such that the face contains all vertices v of the cube such that $v_i = z_i$ for all i with $z_i \neq *$. We call such a face *characterized* by the set $\{i \mid z_i \neq *\}$.

Next, consider the following embedding of the cube into $\mathbb{R}^d$: Each vertex v in $\{-1,1\}^d$ is mapped to the point x_s with $s \in \{+,-\}^d$ such that $v_i = -1$ if and only if $s_i = -$ for $i \in [d]$. Then as we already argued above, for every face of the cube characterized by a set $I \subseteq [d]$, all its vertices are in a common unbounded cell of $\mathcal{F}_I$, so we can extend the embedding defined on the facets of a face to the entire face. After applying a homeomorphism on $\mathbb{R}^d$ that maps this embedding to the cube $[-1,1]^d \subset \mathbb{R}^d$ we can thus assume that all oriented pseudo-hyperplanes of the color class H_i have the facet $x_i = -1$ on their negative side, and the facet $x_i = 1$ on their positive side.

Consider now the α_i-level L of some color class H_i. We define a function f_i for which $f_i(x) = 0$ if x is on L, $f_i(x) > 0$ if x is above L and $f_i(x) < 0$ if x is below L. To achieve this, set $f_i(x) = 0$ for all points x that lie on L. For all other points x, let $d(x, L)$ denote the distance of x to L. If x is below L, we set $f_i(x) = -d(x, L)$, and if x is above L, we set $f_i(x) = d(x, L)$. Since the distance is a continuous function, f_i is continuous.

By construction, the functions f_i satisfy the conditions of the Poincaré-Miranda theorem. We thus get that there is a point x^* in $[-1,1]^d$ for which $f_i(x^*) = 0$ for all $i \in \{1,\ldots,d\}$. Recall that by the definition of a generalized arrangement, no point can lie on $d+1$ of our pseudo-hyperplanes. Thus, x^* is a point lying on exactly one and above exactly $\alpha_i - 1$ oriented pseudo-hyperplanes of color i for all $i \in [d]$, as required.

Note that x^* lies on one oriented pseudo-hyperplane of each color class. If the arrangement is a rainbow arrangement, then there are only $n_1 \cdot \ldots \cdot n_d$ many possible locations for x^*. As each of these points can only be the solution for one α-vector and there are $n_1 \cdot \ldots \cdot n_d$ different α-vectors, it follows that for each α-vector the solution must be unique. ◀

By the famous topological representation theorem of Folkman and Lawrence [14], Theorem 4 also has implications for oriented matroids. To explain this, we will assume familiarity with oriented matroids as presented for example in the Handbook of Discrete and Computational Geometry [22]. In particular, we consider uniform oriented matroids of rank $d+1$. We denote such an oriented matroid by $M = (E, \mathcal{L})$, where E is the ground set and $\mathcal{L} \subseteq \{-, 0, +\}^E$. Coloring E corresponds to a partition into classes (colors) as $E = E_1 \sqcup \cdots \sqcup E_d$. If we use $c(e)$ to denote the color of element $e \in E$, then we take well-separation to mean that for all $s \in \{-, +\}^d$, there exists a covector $v \in \mathcal{L}$ with $v_e = s_{c(e)}$ for all $e \in E$. By applying the Folkman-Lawrence representation theorem and then Theorem 4, we thus get the following corollary in this setting.

▶ **Corollary 11** (α-Ham-Sandwich for Oriented Matroids). *Let $M = (E, \mathcal{L})$ be a uniform oriented matroid of rank $d+1$ in covector representation with $\mathcal{L} \subseteq \{-, 0, +\}^E$. Assume that M is well-separated, and in particular, E is partitioned into d classes $E = E_1 \sqcup \cdots \sqcup E_d$ (colors). Then for every $\alpha \in [|E_1|] \times \cdots \times [|E_d|]$, there is a covector $v^\alpha \in \mathcal{L}$ with exactly one zero and exactly $(\alpha_i - 1)$ minuses on the subset E_i, for all $i \in [d]$.*

4 Generalizing Well-Separation

Using the Poincaré-Miranda theorem, we can prove an even more general statement by relaxing the definition of well-separation. (Recall the definition of a k-level of a set of oriented pseudo-hyperplanes, as defined in Section 3.3.)

▶ **Definition 12** ((β, γ)-separated). *Let $\mathcal{F} = (H_1, \ldots, H_d)$ be a colored generalized arrangement in $\mathbb{R}^d$, where each color class H_i has size n_i. Let $\beta = (\beta_1, \ldots, \beta_d)$ and $\gamma = (\gamma_1, \ldots, \gamma_d)$ be vectors such that $\beta_i, \gamma_i \in [n_i]$ and $\beta_i \leq \gamma_i$ for all $i \in [d]$. The colored generalized arrangement $\mathcal{F}$ is called* (β, γ)-separated *iff for every sign vector $s \in \{+, -\}^d$, there exists a point $x_s \in \mathbb{R}^d$ such that*

- *x_s lies in an unbounded cell,*
- *x_s lies above the γ_i-level of H_i iff $s_i = +$,*
- *and x_s lies below the β_i-level of H_i iff $s_i = -$,*

for all $i \in [d]$.

Note that by setting $\beta = (1, \ldots, 1)$ and $\gamma = (n_1, \ldots, n_d)$ we recover the definition of well-separated.

▶ **Theorem 13.** *Let $\mathcal{F} = (H_1, \ldots, H_d)$ be a (β, γ)-separated colored generalized arrangement where each color class H_i has size n_i. Then for every $(\alpha_1, \ldots, \alpha_d) \in \{\beta_1, \ldots, \gamma_1\} \times \cdots \times \{\beta_d, \ldots, \gamma_d\}$, there is a point $x_\alpha \in \mathbb{R}^d$ lying on one and above exactly $\alpha_i - 1$ oriented pseudo-hyperplanes of color i for all $i \in [d]$.*

Proof. By assumption $\mathcal{F}$ is (β, γ)-separated, so for each vector $s \in \{+, -\}^d$ there is a point x_s in an unbounded cell that lies above or below all relevant levels of a color class as stated in Definition 12. As the arrangement has finitely many cells, we can place a large enough ball B that contains all the bounded cells. Observe that we can assume x_s to lie outside of B. Consider a subset $I \subseteq [d]$ and a family of vectors $s^1, \ldots, s^k$ for which $s^1_i = \ldots = s^k_i$ for all $i \in I$. Let $\mathcal{F}_\mathcal{I}$ be the union of all relevant levels, that is, the α_i-levels for $\alpha_i \in \{\beta_i, \ldots, \gamma_i\}$, of the color classes $i \in I$. We claim that the points $x_{s^1}, \ldots, x_{s^k}$ lie in a common unbounded cell of $\mathcal{F}_\mathcal{I}$. Indeed, assume for the sake of contradiction that two of the points x_{s^i} and x_{s^j} lie in different unbounded cells. Then there is a level of $\mathcal{F}_\mathcal{I}$ separating them outside of B. However, by definition, x_{s^i} and x_{s^j} must lie on the same side of any relevant level in $\mathcal{F}_\mathcal{I}$.

We thus indeed have that all the points $x_{s^1}, \dots, x_{s^k}$ lie in a common unbounded cell of $\mathcal{F}_{\mathcal{I}}$ outside of B. Now we use the same argument as in the proof of Theorem 4. In particular, we can again find an embedding of the cube $[-1,1]^d$ where each face F_I characterized by a subset $I \subseteq [d]$ lies in this common unbounded cell of $\mathcal{F}_{\mathcal{I}}$. After applying a homeomorphism on $\mathbb{R}^d$ that maps this embedding to the cube $[-1,1]^d \subset \mathbb{R}^d$ we can thus assume that all relevant levels of the color class H_i have the facet $x_i = -1$ on their negative side, and the facet $x_i = 1$ on their positive side.

We now take the same functions f_i as in the proof of Theorem 4, which satisfy the conditions of the Poincaré-Miranda theorem. We thus get that there is a point x^* in $[-1,1]^d$ for which $f_i(x^*) = 0$ for all $i \in \{1, \dots, d\}$. ◀

We think that it is instructive to translate Theorem 13 back to the original setting of colored point sets in $\mathbb{R}^d$ (i.e., the primal setting). Note that in Definition 12, the condition that x_s must lie in an unbounded face has technical reasons: In the proof of Theorem 13, it allows us to easily embed the cube that we need for the Poincaré-Miranda theorem. However, we can get away without this assumption in the case of (straight) hyperplanes: In that case, the k-levels are guaranteed to be connected and piecewise linear, and thus we can embed the cube without assuming that the points x_s lie in unbounded faces. Therefore, when going to the primal setting, we can omit this technical assumption and define (β, γ)-separation for point sets as follows.

▶ **Definition 14.** *Let $\mathcal{P} = (P_1, \dots, P_d) \subset \mathbb{R}^d$ be in weak general position and let $\beta, \gamma \in [|P_1|] \times \cdots \times [|P_d|]$ with $\beta_i \leq \gamma_i$ for all $i \in [d]$ be arbitrary. We say that $\mathcal{P}$ is (β, γ)-separated if for every sign vector $s \in \{+,-\}^d$, there exists a hyperplane h_s such that*

- *h_s lies strictly above at least γ_i points of P_i iff $s_i = +$,*
- *and h_s lies above at most $\beta_i - 1$ points of P_i iff $s_i = -$,*

for all $i \in [d]$.

By dualizing the point set and applying Theorem 13 for the dual hyperplanes, we thus get the following corollary for (β, γ)-separated point sets.

▶ **Corollary 15.** *Let $\mathcal{P} = (P_1, \dots, P_d) \subset \mathbb{R}^d$ be (β, γ)-separated for some $\beta, \gamma \in [|P_1|] \times \cdots \times [|P_d|]$ and in general position[6]. Then for every $(\alpha_1, \dots, \alpha_d) \in \{\beta_1, \dots, \gamma_1\} \times \cdots \times \{\beta_d, \dots, \gamma_d\}$, there exists a unique $(\alpha_1, \dots, \alpha_d)$-cut.*

5 Rainbow Arrangements and Bicolored Stretchability

In this section, we take a closer look at well-separated rainbow arrangements in $\mathbb{R}^2$, where pseudo-hyperplanes are also called pseudo-lines. Concretely, such an arrangement consists of well-separated blue and red oriented pseudo-lines such that every blue and red pseudo-line intersect (and thus cross) exactly once.

An interesting question is whether a given rainbow arrangement can be realized using (straight) lines. In the following, we will prove that deciding this is $\exists\mathbb{R}$-complete. Concretely, we prove that the following formalization of what we call *bicolored stretchability* is $\exists\mathbb{R}$-complete.

[6] We require general position (no $d+1$ points on a common hyperplane) to ensure that the dual arrangement is indeed a generalized arrangement. However, one could easily adapt our proofs to also make this work for weak general position.

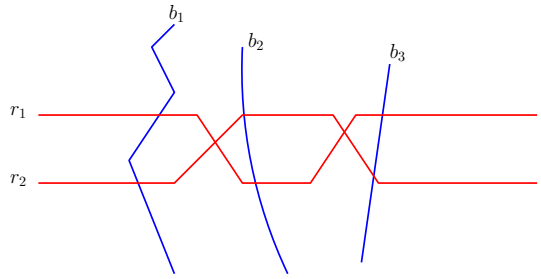


Figure 5 A well-separated rainbow arrangement in $\mathbb{R}^2$. Red pseudo-lines are oriented from left to right, blue pseudo-lines from top to bottom. Note that in order to intersect the blue pseudo-lines in this order, the two red pseudo-lines need to cross twice. In particular, this means that there is no combinatorially equivalent arrangement using straight lines, i.e., this is a NO-instance of bicolored stretchability.

▶ **Definition 16** (Bicolored Stretchability). *Given a combinatorial description of a well-separated rainbow arrangement in $\mathbb{R}^2$ (by specifying for each blue pseudo-line, the order in which it crosses the red pseudo-lines, and vice versa),* bicolored stretchability *is the problem of deciding whether there exists a two-colored well-separated line arrangement with the same combinatorial description.*

In particular, in a YES-instance of bicolored stretchability, all colorful crossings along every line have to happen in the prescribed order.

Observe that, given an instance of bicolored stretchability and a guess for the line arrangement (specifying each line and its color), we can efficiently verify (in a real-RAM machine) whether this guess indeed describes a well-separated line arrangement that is combinatorially equivalent to the bicolored stretchability instance (well-separation can be checked e.g. by comparing the slopes of the lines). Erickson, van der Hoog, and Miltzow [13] conveniently proved that such a verification algorithm on a real-RAM machine is enough to prove membership in $\exists\mathbb{R}$.

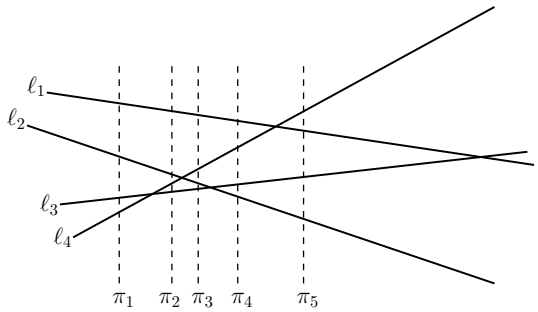


Figure 6 A YES-instance of allowable sequences. The sequence of permutations $\pi_1, \ldots, \pi_5$ can be found in the arrangement by sweeping from left to right.

In order to prove $\exists\mathbb{R}$-hardness, we will provide a reduction from the $\exists\mathbb{R}$-complete problem *allowable sequences* [17]. This problem can be formulated both in its primal form (using point sets) as well as its dual form (using line arrangements). For more details regarding this duality, see e.g., [16]. The following dual formulation will be convenient for our purpose.

▶ **Definition 17** (Allowable Sequences). *Given a sequence of permutations $\pi_1, \ldots, \pi_k$ of $[n]$, where π_{i+1} differs from π_i by a swap of two adjacent items for all $i \in [k-1]$,* allowable sequences *is the problem of deciding whether there exists a line arrangement of n lines such that sweeping the arrangement from left to right and recording how the relative order of the lines changes yields a sequence of permutations that contains the sequence $\pi_1, \ldots, \pi_k$.*

Consider now such a sequence $\pi_1, \ldots, \pi_k$ of permutations of $[n]$, i.e., an instance of allowable sequences. We build an instance of bicolored stretchability as follows: We introduce $n+4$ red pseudo-lines $r_{-1}, r_0, r_1, \ldots, r_n, r_{n+1}, r_{n+2}$ and $2k$ blue pseudo-lines $b_1, \ldots, b_k$ and $b'_1, \ldots, b'_k$. We call the four red lines $r_{-1}, r_0, r_{n+1}, r_{n+2}$ control lines. For each $i \in [k]$, we require both b_i and b'_i to first intersect r_{-1}, r_0 in order, then $r_1, \ldots, r_n$ in the order of the permutation π_i, and finally r_{n+1} and r_{n+2} in order. Conversely, for each $i \in [n]$, we require r_i to intersect the $2k$ blue pseudo-lines in the order $b_1, b'_1, b_2, b'_2, \ldots, b_k, b'_k$. Finally, the control lines are specified such that

- r_{-1} intersects the blue pseudo-lines in the order $b'_1, b_1, \ldots, b'_k, b_k$,
- r_0 intersects the blue pseudo-lines in the order $b_1, b'_1, \ldots, b_k, b'_k$,
- r_{n+1} has to first intersect $b_1, \ldots, b_k$ and then $b'_1, \ldots, b'_k$ in that order,
- and r_{n+2} has to first intersect $b_k, \ldots, b_1$ and then $b'_k, \ldots b'_1$.

Clearly, this construction can be implemented in polynomial time and it remains to prove correctness. However, before getting to the correctness proof, we first observe that this indeed yields a combinatorial description of a rainbow arrangement.

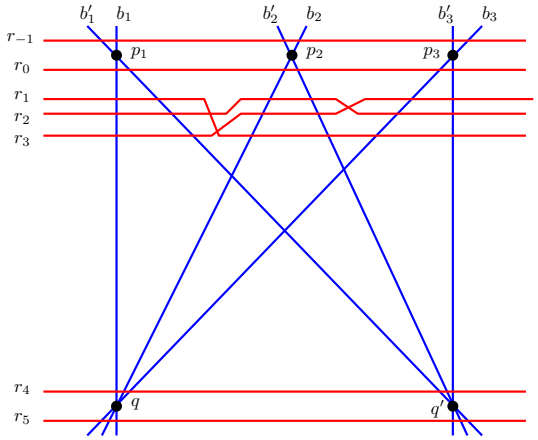


Figure 7 An illustration of the proof of Lemma 18 with $k = n = 3$. Concretely, r_{-1}, r_0, r_4, r_5 are the four control lines.

▶ **Lemma 18.** *There exists a well-separated rainbow arrangement with the combinatorial description described in the construction above.*

Proof. This proof is illustrated in Figure 7. We start by drawing the four control lines as straight horizontal lines in the order (top to bottom) $r_{-1}, r_0, r_{n+1}, r_{n+2}$, making sure that the pairs r_{-1}, r_0 and r_{n+1}, r_{n+2} are relatively close to each other while there is a big gap between r_0 and r_{n+1}. Next, we place k points $p_1, \ldots, p_k$ with increasing x-coordinates between r_{-1} and r_0, and we place two points q and q' with increasing x-coordinates between r_{n+1} and r_{n+2}. The blue lines b_i and b'_i are then drawn as straight lines where b_i goes through p_i and q, and b'_i goes through p_i and q' for all $i \in [k]$. It remains to draw $r_1, \ldots, r_n$ as pseudo-lines

just below r_0. Note that just below r_0, the blue lines read as $b'_1, b_1, \ldots, b'_k, b_k$ from left to right, which is what we need for our red pseudo-lines. Clearly, the red pseudo-lines can be drawn such as to ensure the proper intersections along each of the blue lines. It is not hard to see that this thus yields a well-separated rainbow arrangement. ◀

Next, we prove that applying the construction to a YES-instance of allowable sequences yields a YES-instance of bicolored stretchability. The construction is similar to the proof of Lemma 18. The difference is that, by our assumption that we are given a YES-instance of allowable sequences, we can now ensure that all red lines are straight as well.

▶ **Lemma 19.** *If the permutation $\pi_1, \ldots, \pi_k$ can be realized by a line arrangement (with straight lines), then the corresponding instance of bicolored stretchability can also be realized with straight lines.*

Proof. We take the line arrangement that realizes $\pi_1, \ldots, \pi_k$ and color all n lines red. Next, choose k points $p_1, \ldots, p_k$ above the arrangement with increasing x-coordinates, corresponding to the permutations $\pi_1, \ldots, \pi_k$. In other words, by walking from p_i vertically downward, we are guaranteed to observe the permutation π_i, for all $i \in [k]$. Moreover, we ensure that we will not observe any crossing of two red lines while walking vertically downward. This means that we could even walk at a very small angle (i.e., almost vertically) and still observe the correct permutation.

Next, we choose two points q, q' below the arrangement such that q is to the left of q'. By moving both q and q' vertically down towards negative infinity, we can guarantee that for all $i \in [k]$, the straight blue line b_i (or b'_i, respectively) drawn through p_i and q (or q', respectively) observes the permutation π_i. It remains to draw the control lines:

- r_{-1} is drawn as a horizontal line just above the points $p_1, \ldots, p_k$,
- r_0 is drawn as a horizontal line just below $p_1, \ldots, p_k$,
- r_{n+1} is drawn as a horizontal line just above q, q',
- and r_{n+2} is drawn horizontally just below q, q'. ◀

It remains to prove the other direction, i.e., that a realization of the constructed instance of bicolored stretchability also implies a realization of the allowable sequences instance.

▶ **Lemma 20.** *Assume that given $\pi_1, \ldots, \pi_k$, the constructed instance of bicolored stretchability can be realized using straight lines. Then there also exists a straight-line realization of the allowable sequences instance $\pi_1, \ldots, \pi_k$.*

Proof. Consider the straight-line realization of the bicolored stretchability instance. By construction of the control lines r_{-1} and r_0, b_i and b'_i have to cross at a point p_i that lies above the lines $r_1, \ldots, r_n$ for all $i \in [k]$. Moreover, we know that the control line r_{n+2} lies below the other lines and intersects the blue lines in the order $b_k, \ldots, b_1, b'_k, \ldots, b'_1$. In particular, let q be an arbitrary point on r_{n+2} that lies in-between b_1 and b'_k.

Next, for each $i \in [k]$, draw a new line b''_i through p_i and q. Observe that b''_i intersects r_{n+2} in q, and in particular q lies between the intersections of r_{n+2} with b'_i and b_i, respectively. Since b_i and b'_i intersect $r_1, \ldots, r_n$ in the same order (given by π_i), so does b''_i. This means that each of the lines b''_i observes the correct permutation, and they all pass through the same point q that lies on r_{n+2}. Thus, we can apply a projective transformation that turns r_{n+2} into the line at infinity, which in turn ensures that the lines $b''_1, \ldots, b''_k$ are parallel. We conclude that the red lines $r_1, \ldots, r_n$ after the transformation are a realization of the original allowable sequences instance. ◀

Putting all of this together, we conclude that bicolored stretchability is indeed $\exists\mathbb{R}$-complete.

▶ **Theorem 21.** *Bicolored stretchability is $\exists\mathbb{R}$-complete.*

Proof. We argued that it is contained in $\exists\mathbb{R}$ by using the verification-approach in [13]. $\exists\mathbb{R}$-hardness follows from our reduction from allowable sequences above, where correctness follows from combining Lemma 19 and Lemma 20. ◀

Certainly, Theorem 21 also implies $\exists\mathbb{R}$-hardness for the realizability problem of more general two-colored rainbow arrangements (that are not necessarily well-separated). Concretely, this also implies that realizability of bicolored order types (that were studied in, e.g., [2]) is $\exists\mathbb{R}$-hard. However, the well-separation condition is useful for us, as it allows us to also conclude that realizability of grid USO[7] is $\exists\mathbb{R}$-complete.

▶ **Corollary 22.** *Deciding whether a given grid USO is realizable is $\exists\mathbb{R}$-complete, even in two dimensions.*

Proof. Note that containment in $\exists\mathbb{R}$ follows again by giving a real-RAM verifier, and we will not go into the details of this. Instead, we explain why USO realizability is $\exists\mathbb{R}$-hard based on prior work: It was proven in [10] that realizable grid USOs correspond to well-separated hyperplane arrangements. This allows us to reduce bicolored stretchability to realizability of two-dimensional grid USO: Given the well-separated rainbow arrangement, construct the orientation as described in Section 2. This must yield a two-dimensional grid USO. If this grid USO is realizable, then by [10] there exists a well-separated line arrangement corresponding to this grid USO, which hence must have the same combinatorial description as the initial well-separated rainbow arrangement. Conversely, any well-separated line arrangement with the same combinatorial information implies that the constructed orientation is realizable. ◀

References

1 Oswin Aichholzer, Nieves Atienza, José M. Díaz-Báñez, Ruy Fabila-Monroy, David Flores-Peñaloza, Pablo Pérez-Lantero, Birgit Vogtenhuber, and Jorge Urrutia. Computing balanced islands in two colored point sets in the plane. *Information Processing Letters*, 135:28–32, July 2018. doi:10.1016/j.ipl.2018.02.008.

2 Oswin Aichholzer and Anna Brötzner. Bicolored Order Types. *Computing in Geometry and Topology*, 3(2):3:1–3:17, 2024. doi:10.57717/cgt.v3i2.46.

3 Arseniy Akopyan and Roman N. Karasev. Cutting the Same Fraction of Several Measures. *Discrete & Computational Geometry*, 49(2):402–410, March 2013. doi:10.1007/s00454-012-9450-4.

4 David Ariza-Ruiz, Jesús Garcia-Falset, and Simeon Reich. The Bolzano-Poincaré-Miranda theorem in infinite-dimensional Banach spaces. *J. Fixed Point Theory Appl.*, 21(2):Paper No. 59, 12, 2019. doi:10.1007/s11784-019-0701-3.

5 Imre Bárány, Alfredo Hubard, and Jesús Jerónimo. Slicing Convex Sets and Measures by a Hyperplane. *Discrete & Computational Geometry*, 39(1):67–75, 2008. doi:10.1007/s00454-007-9021-2.

6 Imre Bárány and Jiří Matoušek. Equipartition of two measures by a 4-fan. *Discrete & Computational Geometry*, 27(3):293–301, 2002. doi:10.1007/S00454-001-0071-6.

[7] A grid USO is called realizable if it is induced by an instance of the *P-Matrix Generalized Linear Complementarity Problem (PGLCP)*. We do not elaborate on the details of realizability of grid USOs here and instead rely on prior work that establishes the appropriate connection with hyperplane arrangements [10].

7 Luis Barba, Alexander Pilz, and Patrick Schnider. Sharing a pizza: bisecting masses with two cuts. *arXiv preprint arXiv:1904.02502*, 2019. arXiv:1904.02502.

8 Pavle V. M. Blagojević and Aleksandra Dimitrijević Blagojević. Using equivariant obstruction theory in combinatorial geometry. *Topology and its Applications*, 154(14):2635–2655, 2007. doi:10.1016/j.topol.2007.04.007.

9 Pavle V. M. Blagojević, Aleksandra Dimitrijević Blagojević, Roman Karasev, and Jonathan Kliem. More bisections by hyperplane arrangements. *Discrete Comput. Geom.*, 67(1):33–64, 2022. doi:10.1007/s00454-021-00337-w.

10 Michaela Borzechowski, John Fearnley, Spencer Gordon, Rahul Savani, Patrick Schnider, and Simon Weber. Two Choices Are Enough for P-LCPs, USOs, and Colorful Tangents. In *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297, pages 32:1–32:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2024.32.

11 Michaela Borzechowski, Sebastian Haslebacher, Hung P. Hoang, Patrick Schnider, and Simon Weber. Splitting sandwiches unevenly via unique sink orientations and rainbow arrangements, 2026. arXiv:2602.10795.

12 Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

13 Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. Smoothing the Gap Between NP and ER. *SIAM Journal on Computing*, 53(6):FOCS20–102, 2024. doi:10.1137/20M1385287.

14 Jon Folkman and Jim Lawrence. Oriented matroids. *Journal of Combinatorial Theory, Series B*, 25(2):199–236, 1978. doi:10.1016/0095-8956(78)90039-4.

15 Bernd Gärtner, Walter D. Morris jr., and Leo Rüst. Unique sink orientations of grids. *Algorithmica*, 51(2):200–235, 2008. doi:10.1007/s00453-007-9090-x.

16 Jacob E. Goodman and Richard Pollack. Allowable Sequences and Order Types in Discrete and Computational Geometry. In *New Trends in Discrete and Computational Geometry*, pages 103–134. Springer, Berlin, Heidelberg, 1993. doi:10.1007/978-3-642-58043-7_6.

17 Udo Hoffmann and Keno Merckx. A universality theorem for allowable sequences with applications. doi:10.48550/arXiv.1801.05992.

18 Alfredo Hubard and Roman Karasev. Bisecting measures with hyperplane arrangements. *Mathematical Proceedings of the Cambridge Philosophical Society*, 169(3):639–647, 2019. doi:10.1017/S0305004119000380.

19 Alfredo Hubard and Pablo Soberón. Bisecting masses with families of parallel hyperplanes. *arXiv preprint arXiv:2404.14320*, 2024.

20 Carlo Miranda. Un'osservazione su un teorema di Brouwer. *Boll. Un. Mat. Ital. (2)*, 3:5–7, 1940.

21 Henri Poincaré. Sur certaines solutions particulières du problème des trois corps. *Bulletin astronomique, Observatoire de Paris*, 1(1):65–74, 1884. doi:10.3406/bastr.1884.9762.

22 Jürgen Richter-Gebert and Günter M. Ziegler. Oriented matroids. In *Handbook of Discrete and Computational Geometry*, pages 111–132. CRC Press, Inc., USA, 1997.

23 Edgardo Roldán-Pensado and Pablo Soberón. A survey of mass partitions. *Bull. Amer. Math. Soc. (N.S.)*, 59(2):227–267, 2022. doi:10.1090/bull/1725.

24 Patrick Schnider. Equipartitions with wedges and cones. *arXiv preprint arXiv:1910.13352*, 2019. arXiv:1910.13352.

25 Patrick Schnider. The Complexity of Sharing a Pizza. In *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, volume 212, pages 13:1–13:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2021.13.

26 Patrick Schnider and Pablo Soberón. Cookie cutters: Bisections with fixed shapes. *Advances in Applied Mathematics*, 171:102957, 2025. doi:10.1016/j.aam.2025.102957.

27 Pablo Soberón and Yuki Takahashi. Lifting Methods in Mass Partition Problems. *International Mathematics Research Notices*, 2023(16):14103–14130, 2023. doi:10.1093/imrn/rnac224.

28 William Steiger and Jihui Zhao. Generalized Ham-Sandwich Cuts. *Discrete & Computational Geometry*, 44(3):535–545, 2010. doi:10.1007/s00454-009-9225-8.

29 Alan Stickney and Layne Watson. Digraph models of Bard-type algorithms for the linear complementarity problem. *Mathematics of Operations Research*, 3(4):322–333, 1978. doi:10.1287/MOOR.3.4.322.

30 A. H. Stone and J. W. Tukey. Generalized "sandwich" theorems. *Duke Math. J.*, 9(2):356–359, June 1942. doi:10.1215/S0012-7094-42-00925-6.

31 Tibor Szabó and Emo Welzl. Unique sink orientations of cubes. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 547–555, 2001. doi:10.1109/SFCS.2001.959931.

The Spanning Ratio of the Directed Θ_6-Graph Is 5

Prosenjit Bose
School of Computer Science, Carleton University, Ottawa, Canada

Jean-Lou De Carufel
School of Electrical Engineering and Computer Science, University of Ottawa, Canada

John Stuart
School of Electrical Engineering and Computer Science, University of Ottawa, Canada

Darryl Hill
School of Computer Science, Carleton University, Ottawa, Canada

Abstract

Given a finite set $P \subset \mathbb{R}^2$, the directed Theta-6 graph, denoted $\vec{\Theta}_6(P)$, is a well-studied geometric graph due to its close relationship with the Delaunay triangulation. The $\vec{\Theta}_6(P)$-graph is defined as follows: the plane around each point $u \in P$ is partitioned into 6 equiangular cones with apex u, and in each cone, u is joined to the point whose projection on the bisector of the cone is closest. Equivalently, the $\vec{\Theta}_6(P)$-graph contains an edge from u to v exactly when the interior of ∇_u^v is disjoint from P, where ∇_u^v is the unique equilateral triangle containing u on a corner, v on the opposite side, and whose sides are parallel to the cone boundaries. It was previously shown that the spanning ratio of the $\vec{\Theta}_6(P)$-graph is between 4 and 7 in the worst case (Akitaya, Biniaz, and Bose *Comput. Geom.*, 105-106:101881, 2022). We close this gap by showing a tight spanning ratio of 5. This is the first tight bound proven for the spanning ratio of any $\vec{\Theta}_k(P)$-graph. Our lower bound models a long path by mapping it to a converging series. Our upper bound proof uses techniques novel to the area of spanners. We use linear programming to prove that among several candidate paths, there exists a path satisfying our bound.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners; Theory of computation → Computational geometry; Theory of computation → Shortest paths

Keywords and phrases Geometric Spanners, Theta Graphs, Directed Theta Graphs, Spanning Ratio, Computational Geometry

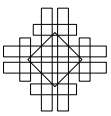
Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.20

Related Version *Full Version*: `https://arxiv.org/abs/2603.09048` [5]

1 Introduction

Studying the distance-preserving properties of particular families of graphs in the plane has been an active area of research for decades. This area has numerous applications, such as motion planning, or wireless routing. We will focus on geometric graphs where each vertex is a point in the plane, and each edge is weighted by the Euclidean distance between its endpoints. Given a geometric graph G, the spanning ratio of a subgraph H is a measure of how well distances of G are preserved. In particular, a subgraph H is called a *c-spanner* of G if for every edge uv in G, the shortest path in H from u to v is at most c times the length of the edge uv in G [15]. The smallest such c is referred to as the *spanning ratio* of H. Finding a tight bound on the spanning ratio of different geometric graphs is a fundamental problem in computational geometry. A particular graph of interest is the *Delaunay triangulation*. It has an edge uv when there exists a disk with u and v on its boundary and no vertices in its interior. The spanning ratio of the Delaunay triangulation has been studied for nearly 40 years, and the first constant upper bound proven was $\frac{1+\sqrt{5}}{2}\pi \approx 5.083$ [12]. Currently, the spanning ratio of the Delaunay triangulation is only known to be between 1.593 [21]


© Prosenjit Bose, Jean-Lou De Carufel, John Stuart, and Darryl Hill;
licensed under Creative Commons License CC-BY 4.0



and 1.998 [20], and finding the exact value remains a major open problem [15]. The first plane geometric graph proven to have a constant spanning ratio is the $\square$-Delaunay graph, whose empty region is a square [9]. The proof was then generalized to the $\triangle$-Delaunay graph, whose empty region is an equilateral triangle [10].

Delaunay triangulations have many desirable properties. A Delaunay triangulation is a plane graph, meaning that it contains roughly $3|V|$ undirected edges, therefore the average vertex degree is approximately 6. However, it is possible for a vertex to have arbitrarily large degree. In many applications, such as signal transmission, such a large degree may be problematic. Instead, we focus on a closely-related geometric graph with bounded outdegree of 6.

Given a finite set $P \subset \mathbb{R}^2$ and integer $k > 1$, we can construct a geometric graph on P with at most k outgoing edges per vertex as follows. For each vertex $u \in P$, partition the plane into k equiangular cones about u. Then, add an edge from u to the *nearest* vertex in each cone about u. Repeat this process for each vertex of P. If *nearest* is measured by the Euclidean distance, then the resulting graph is known as a Yao-k graph, first studied by Yao [22]. The directed Theta-k graph, denoted $\vec{\Theta}_k$, is defined similarly, except that *nearest* is measured using a distance function whose unit disk is a regular k-gon. In particular, the $\vec{\Theta}_k$-graph contains an edge from u to v in cone C when the orthogonal projection of uv onto the cone bisector is minimal among all vertices v in cone C. We refer to the undirected version as the Θ_k-graph. Since being introduced by Clarkson [11] and Keil and Gutwin [13], Θ_k-graphs have been studied extensively. For $k \geq 4$, the Θ_k-graph is known to be a spanner of the complete graph [2, 4, 6, 11, 13, 17]. On the other hand, the $\vec{\Theta}_k$-graph is known to be a constant spanner for $k \geq 6$ [1, 6, 17] and also for $k = 4$ [4]. Despite extensive research, tight bounds on the spanning ratio are only known for the Θ_{4k+2}-graph when $k \geq 1$. In general, results claiming tight bounds on the spanning ratio of any geometric graph are rare. To the best of our knowledge, the only other geometric graphs with known tight spanning ratios are the three generalized Delaunay graphs whose empty regions are either a triangle, square, or hexagon. See Table 1.

Table 1 Other geometric graphs with known tight spanning ratios.

Reference	Graph	Spanning Ratio	Notes
[6]	Θ_{4k+2} $(k \geq 1)$	$1 + 2\sin(\theta/2)$	Also for constrained version [8]
[10]	$\triangle$-Delaunay	2	Also for affine transformations [18, 14]
[3]	$\square$-Delaunay	$\sqrt{4+2\sqrt{2}} \approx 2.613$	Also for affine transformations [7, 19]
[16]	$\hexagon$-Delaunay	2	

The $\vec{\Theta}_6$-graph has a special symmetry. For two points $u, v \in P$, we can define ∇_u^v to be the unique equilateral triangle whose sides have slopes $-\sqrt{3}$, 0, and $\sqrt{3}$ and u is on a corner and v is on the opposite side. Then the $\vec{\Theta}_6$-graph contains an edge from u to v exactly when the interior of ∇_u^v contains no vertices of P. Readers familiar with Delaunay triangulations may notice that the $\vec{\Theta}_6$-graph is essentially two rotated copies of a $\triangle$-Delaunay graph. The $\triangle$-Delaunay graph, often referred to as the TD-Delaunay graph, has a spanning ratio of 2 in the worst case, hence the Theta-6 graph inherits this upper bound. On the other hand, the $\vec{\Theta}_6$-graph does not preserve distances nearly as well as its undirected counterpart. The first bounds on the spanning ratio of the $\vec{\Theta}_6$-graph were given by Akitaya, Biniaz, and Bose [1]. For any $\epsilon > 0$, they construct a $\vec{\Theta}_6$-graph whose spanning ratio is at least $4 - \epsilon$. Then, they show that constructing a path in an arbitrary $\vec{\Theta}_6$-graph is not as straightforward as selecting

the outgoing edge in the cone containing the destination. This technique is known as *greedy routing*, and it produces efficient paths in $\vec{\Theta}_k$-graphs only when $k \geq 7$ [17]. Greedy routing in $\vec{\Theta}_6$-graphs can lead to a spiraling path of unbounded length. Instead, the authors of [1] give an upper bound of 7 for the spanning ratio of $\vec{\Theta}_6$-graphs using the following technique. Given an edge from a to b, they show that the *greedy* path from b to a has length at most 6 times the length of the edge from a to b. Then, for any two vertices s, t, it suffices to find an intermediate vertex x such that there are short paths from s to x and t to x. Indeed, a path from s to t can be formed by concatenating the path from s to x with the greedy paths corresponding to each edge on the path from t to x. The authors leave it as an open problem to close the gap between 4 and 7. As the title of our paper suggests, we close this gap by showing that the $\vec{\Theta}_6$-graph has a tight spanning ratio of 5.

Our proof uses elements from [1] along with several novel techniques in the area of spanners. Most notably, we use linear programming to prove the existence of a short path. At a high level, our proof is an induction on the distance between points, and it proceeds as follows. We define a special region R between s and t, and show that if there exists a vertex $x \in R$, then induction provides two paths, from s to x and from x to t, which can be concatenated to form a sufficiently short path from s to t. Then, for the rest of the proof, we assume that no vertices are in the interior of R. Importantly, we are able to show that any edge uv on the greedy path that crosses R must pay a "toll". More precisely, v is guaranteed to be at least twice as close to t as u. We leverage this property by applying induction after a candidate path crosses R. The difficulty is that there are multiple candidate paths, and determining which path is sufficiently short is non-trivial. Our solution is to assume that there is no sufficiently short candidate path. This allows us to prove inequalities corresponding to each candidate path, then we can use linear programming to show that the system is infeasible. In essence, assuming that there is no short path leads to a contradiction.

2 Preliminaries

The geometric graphs we study are weighted, directed graphs whose vertex sets contain points in the plane. We denote the directed edge from vertex u to vertex v by uv, and with a slight abuse of notation, when $p, q \in \mathbb{R}^2$, we denote the segment from p to q as pq. In general, for an edge uv and a norm $\|\cdot\|$, the value $\|uv\|$ is exactly the factor by which the unit disk of $\|\cdot\|$ must be scaled such that the unique translate with u in the center has v on the boundary. For example, $\|\cdot\|_2$ refers to the Euclidean norm whose unit disk is a circle with radius 1. We also let $\|uv\|_x$ (resp. $\|uv\|_y$) denote the absolute difference in x-coordinate (resp. y-coordinate) from u and v. The *Manhattan norm*, $\|uv\|_1 := \|uv\|_x + \|uv\|_y$, has a tipped square with side length $\sqrt{2}$ as its unit disk. For an edge uv in a geometric graph, we define its weight to be $\|uv\|_2$, the usual distance from u to v. Let $\|\cdot\|_{\hexagon}$ denote the *hexagonal norm* whose unit disk is a regular hexagon with east and west vertices equal to $(1, 0)$ and $(-1, 0)$, respectively. On the other hand, let $\|\cdot\|_{\bigcirc}$ denote the *tipped hexagonal norm* whose unit disk is a tipped regular hexagon with north and south vertices equal to $(0, \frac{2}{\sqrt{3}})$ and $(0, \frac{-2}{\sqrt{3}})$, respectively. See Figure 1. Notice that we always have $\|uv\|_{\bigcirc} \leq \|uv\|_2 \leq \|uv\|_{\hexagon}$, and equality holds exactly when uv has slope $-\sqrt{3}$, 0, or $\sqrt{3}$.

In a geometric graph G, we define the length of a path to be the sum of the weights of its edges. For any norm $\|\cdot\|$ and path $\mathcal{P}$, we define $\|\mathcal{P}\| := \sum_{uv \text{ on } \mathcal{P}} \|uv\|$. If v is a vertex on a path $\mathcal{P}$, then we define the *truncated path* $\mathcal{P}_v$ as the subpath of $\mathcal{P}$ which ends at v. For two vertices u, v in a geometric graph G, we let $d_G(u, v)$ denote the length of the shortest path in G from u to v. Recall that all edges are weighted by the Euclidean norm.

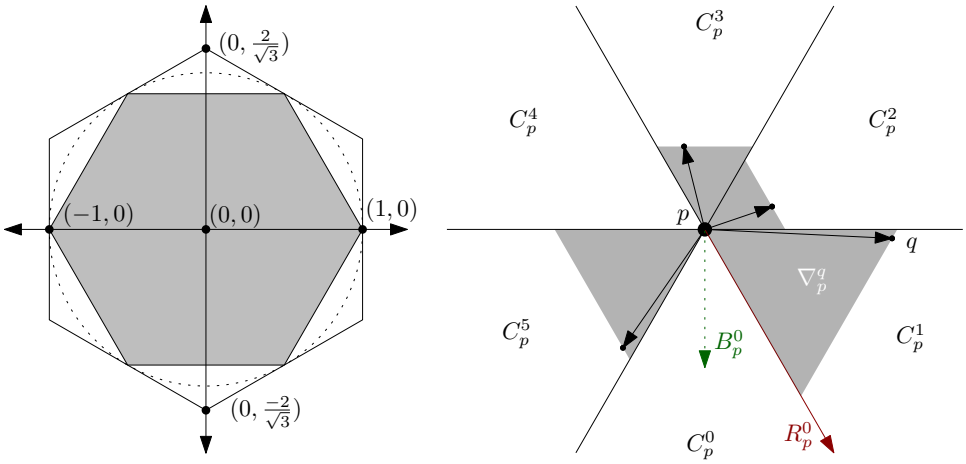


Figure 1 Left: The dotted circle represents the unit disk in the standard Euclidean norm ($\|\cdot\|_2$). The shaded inner hexagon is the unit disk in the $\|\cdot\|_{\hexagon}$-norm. The outer tipped hexagon is the unit disk in the $\|\cdot\|_{\pentagon}$-norm. Right: The six cones about p are labeled in counterclockwise order from C_p^0 below p. The dotted green bisector is B_p^0 and the red ray is R_p^0. The empty shaded triangle in C_p^1 is ∇_p^q. Each shaded region is empty.

Then for a constant $c \geq 1$, G is said to be a c-spanner if for all points u, v in G, we have $d_G(u, v) \leq c\|uv\|_2$. The *spanning ratio* of G is the least c for which G is a c-spanner. The spanning ratio of a class of graphs $\mathcal{G}$ is the least c for which all graphs in $\mathcal{G}$ are c-spanners.

For any point p, consider the three lines passing through p with slopes $-\sqrt{3}$, 0, and $\sqrt{3}$. These lines partition the plane into six cones about p, which we label counter-clockwise by C_p^i for $i \in \mathbb{Z}_6$ starting with C_p^0 directly below p. Here, $\mathbb{Z}_6$ refers to the integers modulo 6. See Figure 1. For each cone C_p^i, we denote its bisector by B_p^i. Throughout this paper, we adopt the convention that cones are closed sets, otherwise we write $\text{int}(R)$ to denote the interior of region R. For $i \in \mathbb{Z}_6$, we define the ray $R_p^i := C_p^i \cap C_p^{i+1}$. For points $p, q \in \mathbb{R}^2$ with $q \in C_p^i$, we let ∇_p^q denote the triangular region $\{r \in C_p^i \mid \|pr\|_{\hexagon} \leq \|pq\|_{\hexagon}\}$. For any equilateral triangle T, we let $\|T\|$ denote the side length of T. Notice that $\|\nabla_p^q\| = \|pq\|_{\hexagon}$. For any finite point set $P \subset \mathbb{R}^2$, we make the *general position assumption* that no two vertices lie on a line with slope $-\sqrt{3}$, 0, or $\sqrt{3}$. The directed Theta-6 graph of P, denoted $\vec{\Theta}_6(P)$, contains an edge from vertex u to vertex v exactly when $\text{int}(\nabla_u^v) \cap P = \emptyset$. Intuitively, a vertex has an outgoing edge to the nearest vertex in each of its six cones, measured using the $\|\cdot\|_{\hexagon}$-norm. For any vertices $u, v \in P$, we let $\pi(u, v)$ denote the *greedy path* from u to v. In particular, every edge ab of $\pi(u, v)$ satisfies $b \in C_a^i \iff v \in C_a^i$. If w is a vertex along the path $\pi(u, v)$, then the subpath terminating at w is denoted by $\pi(u, v)_w$. Note that if $w \neq v$, then $\pi(u, v)_w$ is not necessarily the same as $\pi(u, w)$.

3 Lower Bound

▶ **Lemma 1.** *For any $\epsilon > 0$, there exists a $\vec{\Theta}_6$-graph with spanning ratio at least $5 - \epsilon$.*

Proof. Let $\epsilon > 0$ be arbitrarily small. We will construct a $\vec{\Theta}_6$-graph with spanning ratio at least $5-\epsilon$, shown in Figure 2. For the following construction, set $0 < \delta < \frac{\epsilon}{72}$. Define the points $t := (0, 0)$, $s := (-1, 0) + \delta(1, -1), a := (-\frac{1}{2}, \frac{\sqrt{3}}{2}) + \delta(1, -2), b := (\frac{1}{2}, \frac{\sqrt{3}}{2}) + \delta(-2, -3), c := (1, 0) + \delta(-6, 1)$. Next, let p^0 be the unique point such that tp^0 has slope $-\delta$ and p^0c has slope $\sqrt{3} - \delta$. Next, let q^0 be on segment ct such that p^0q^0 has slope $\delta - \sqrt{3}$. For $i > 0$, let

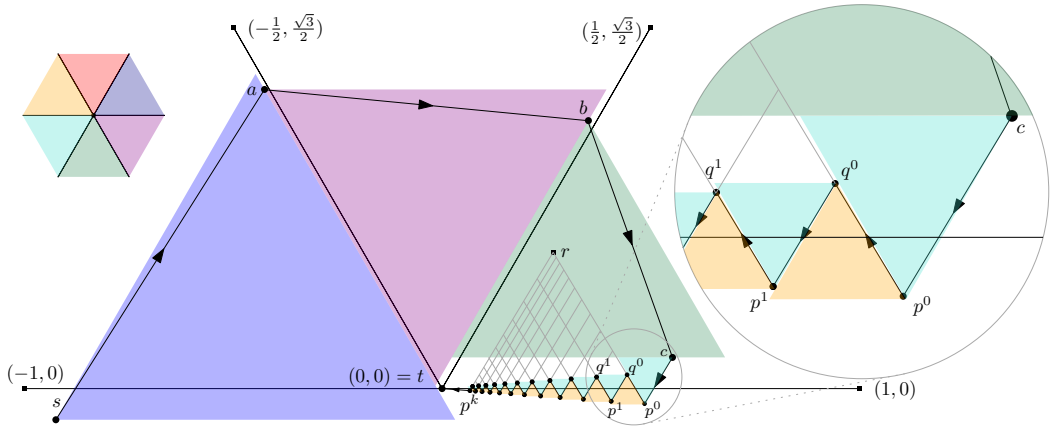


Figure 2 Graph construction achieving a spanning ratio lower bound of $5 - \epsilon$. The shortest path from s to t is the greedy path $s, a, b, c, p^0, q^0, p^1, q^1, ..., , q^{k-1}, p^k, t$. Top Left: The colour coding for edges based on direction.

q^i be on ct such that $\frac{\|q^{i-1}q^i\|_2}{\|tq^i\|_2} = \frac{\|q^0c\|_2}{\|tc\|_2}$. For $i > 0$, also define p^i to be on tp^0 such that p^iq^i has slope $\delta - \sqrt{3}$. We let k be the least integer such that p_k has x-coordinate less than δ. We have constructed the point set $P := \{t, s, a, b, c\} \cup \{p^0, p^1, ..., p^k\} \cup \{q^0, q^1, ..., q^{k-1}\}$. In $\vec{\Theta}_6(P)$, the shortest path from s to t is $s, a, b, c, p^0, q^0, p^1, q^1, ...q^{k-1}, p^k, t$ since from the perspective of any vertex on the path, all subsequent vertices lie in the same cone. Now we will lower bound the length of this path. First, we have

$$
\begin{aligned}
1 &= \|(-1,0)(-\frac{1}{2}, \frac{\sqrt{3}}{2})\|_2 && \text{since } \sqrt{\left(\frac{1}{2}\right)^2 + \left(\frac{\sqrt{3}}{2}\right)^2} = 1, \\
&\leq \|(-1,0)s\|_2 + \|sa\|_2 + \|a(-\frac{1}{2}, \frac{\sqrt{3}}{2})\|_2 && \text{by triangle inequality,} \\
&\leq \|(-1,0)s\|_1 + \|sa\|_2 + \|a(-\frac{1}{2}, \frac{\sqrt{3}}{2})\|_1 && \text{since } \|\cdot\|_2 \leq \|\cdot\|_1, \\
&= 2\delta + \|sa\|_2 + 3\delta && \text{by definition.} \qquad (1)
\end{aligned}
$$

which implies $\|sa\|_2 \geq 1 - 5\delta$. Using the same technique, we can obtain the following bounds: $\|ab\|_2 \geq 1 - 8\delta$ and $\|bc\|_2 \geq 1 - 12\delta$. Next, let r be the unique point such that p^kr has slope $\sqrt{3} - \delta$ and rp^0 has slope $\delta - \sqrt{3}$. Then by unfolding, notice that the path $p^0, q^0, p^1, q^1, ..., p^{k-1}, q^{k-1}, p^k$ has length $\|p^0r\|_2 + \|rp^k\|_2$. See Figure 2. By the triangle inequality, we have $\|p^0(1,0)\|_1 \leq \|p^0c\|_x + \|p^0c\|_y + \|c(1,0)\|_1 \leq 2\delta + 2\delta + 7\delta = 11\delta$ and $\|tp^k\|_1 = \|tp^k\|_x + \|tp^k\|_y \leq \delta + \delta = 2\delta$. Next, we claim that r approaches $(\frac{1}{2}, \frac{\sqrt{3}}{2})$ as $\delta \to 0$.

▷ Claim 2. We have $\|r(\frac{1}{2}, \frac{\sqrt{3}}{2})\|_2 \leq 17\delta$. See full version for details [5].

Now we can apply the same technique used to prove inequality (1) to obtain $\|p^0r\|_2 \geq 1 - 28\delta$ and $\|rp^k\|_2 \geq 1 - 19\delta$. Therefore the total length of the path is at least

$$
\begin{aligned}
d_{\vec{\Theta}_6(P)}(s,t) &= \|sa\|_2 + \|ab\|_2 + \|bc\|_2 + \|cp^0\|_2 + \left(\sum_{i=0}^{k-1} \|p^iq^i\|_2\right) + \|q^{k-1}p^k\|_2 + \|p_kt\|_2 \\
&\geq (1 - 5\delta) + (1 - 8\delta) + (1 - 12\delta) + (1 - 28\delta) + (1 - 19\delta) \\
&= 5 - 72\delta > 5 - \epsilon.
\end{aligned}
$$

Then since $\|st\|_2 \leq 1$, the spanning ratio of $\vec{\Theta}_6(P)$ is at least $5 - \epsilon$. ◀

4 Upper Bound

The goal of this section is to prove Theorem 3.

▶ **Theorem 3.** *For any point set $P \subset \mathbb{R}^2$ in general position, we have $d_{\vec{\Theta}_6(P)}(s,t) \leq 5\|st\|_{\bigcirc} \leq 5\|st\|_2$ $\forall s,t \in P$.*

Let $P \subset \mathbb{R}^2$ be a point set in general position, and let $G := \vec{\Theta}_6(P)$. Recall that the $\|\cdot\|_{\hexagon}$-norm is used to define G, however Theorem 3 is expressed in terms of the $\|\cdot\|_{\bigcirc}$-norm. The following lemma describes the exact relationship between $\|\cdot\|_{\hexagon}$ and $\|\cdot\|_{\bigcirc}$.

▶ **Lemma 4.** *Let $u \in \mathbb{R}^2$, $i \in \mathbb{Z}_6$, and $v \in C_u^i$. Then*

$$\|uv\|_{\hexagon} = \|uv\|_{\bigcirc} + \frac{1}{2}\min(\|C_u^i \cap C_v^{i-2}\|, \|C_u^i \cap C_v^{i+2}\|).$$

Proof. Suppose without loss of generality that $i = 0$ and v is to the right of the bisector B_u^0. See Figure 3. Then $\|C_u^0 \cap C_v^2\| \leq \|C_u^0 \cap C_v^4\|$. Let a be the orthogonal projection of v on R_u^0. Also let b be the intersection of R_u^0 with a horizontal line through v. Then

$$\|uv\|_{\bigcirc} + \frac{1}{2}\|C_u^0 \cap C_v^2\| = \|ua\|_2 + \|ab\|_2 = \|ub\|_2 = \|uv\|_{\hexagon}.$$

The other cases follow by symmetry. ◀

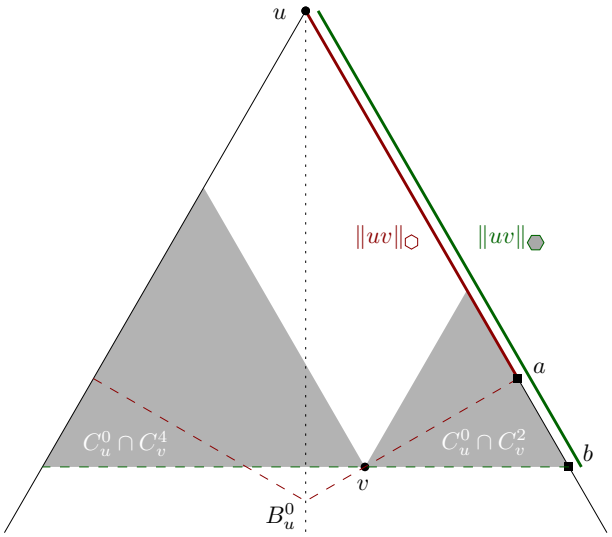


Figure 3 Lemma 4 with $v \in C_u^0$. The dotted vertical ray is the bisector B_u^0. The horizontal green dashed segment represents all points $p \in C_u^0$ such that $\|up\|_{\hexagon} = \|uv\|_{\hexagon}$. The red dashed segments represent all points $p \in C_u^0$ such that $\|up\|_{\bigcirc} = \|uv\|_{\bigcirc}$.

The next lemma finds use in proving Theorem 3

▶ **Lemma 5.** *Let $u, v, t \in P$ and $i \in \mathbb{Z}_6$. Suppose $u, v \in C_t^i \cup C_t^{i+1}$ between B_t^i and B_t^{i+1}. Then if $v \in C_u^{i+3} \cup C_u^{i+4}$, we have $\|uv\|_{\hexagon} \leq 2(\|ut\|_{\bigcirc} - \|vt\|_{\bigcirc})$.*

Proof. Without loss of generality, suppose $i = 0$. If $v \in C_u^3$, then define $p := R_u^2 \cap R_v^1$, otherwise, if $v \in C_u^4$, then define $p := R_u^4 \cap R_v^5$. Let v' and p' be the orthogonal projections of v and p on R_u^3. See Figure 4. Then

$$\|uv\|_{\hexagon} = \|up\|_2 = 2\|up'\|_2 \leq 2\|uv'\|_2 = 2(\|ut\|_{\bigcirc} - \|vt\|_{\bigcirc}).$$ ◀

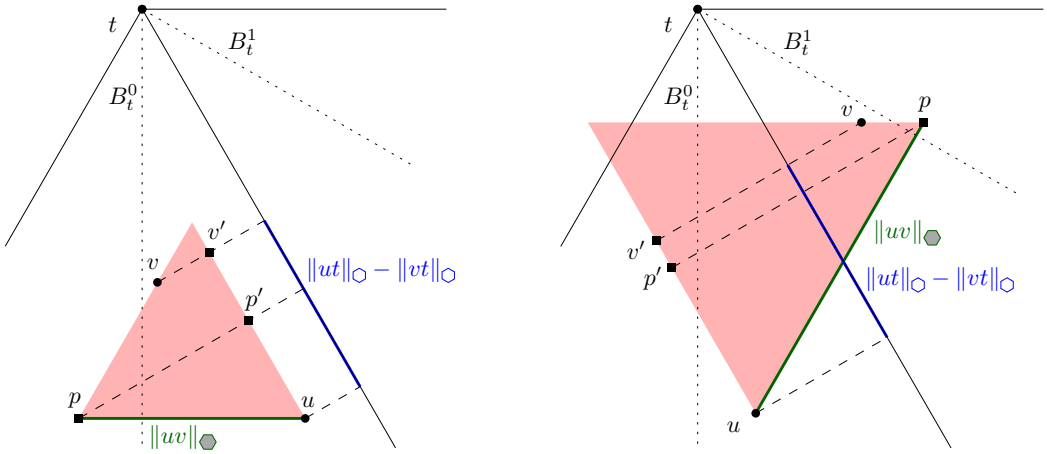


Figure 4 For the proof of Lemma 5, we assume $i = 0$ and show the case when $v \in C_u^4$ on the left and $v \in C_u^3$ on the right.

We prove Theorem 3 by induction on the hexagonal distance $\|\cdot\|_{\hexagon}$ between vertices. For the base case, consider the closest pair of vertices $s, t \in P$ (i.e. $\|st\|_{\hexagon}$ is minimal). Minimality implies $\text{int}(\nabla_s^t)$ must not contain any vertices of P. Therefore there must be an edge from s to t, yielding $d_G(s,t) = \|st\|_2 \leq 5\|st\|_{\bigcirc}$.

For the remainder of the proof, fix $s, t \in P$ and suppose that for any vertices $u, v \in P$, if $\|uv\|_{\hexagon} < \|st\|_{\hexagon}$, then $d_G(u,v) \leq 5\|uv\|_{\bigcirc}$. We will assume $\|st\|_{\hexagon} = 1$ since our argument holds when scaling the point set. Without loss of generality, suppose $s \in C_t^0$, with s being to the right of B_t^0. Define the triangle $T = C_t^0 \cap C_s^2$, and denote its side length by $y_0 := \|T\|$. Notice that y_0 describes the *hex* distance from s to the nearest boundary, and by Lemma 4, we obtain $5\|st\|_{\bigcirc} = 5 - 2.5y_0$. Suppose towards a contradiction that $5\|st\|_{\bigcirc} < d_G(s,t)$. Our approach is to analyse paths from s to t, which will allow us to prove inequalities based on our faulty assumption. These inequalities will form a linear system, then linear programming will help us show that the system is infeasible. The first step in our proof is to establish an empty region which we refer to as R.

▶ **Definition 6** (Empty Region R). *Consider the points $p := B_s^3 \cap B_t^1$ and $q = B_s^4 \cap B_t^0$. See Figure 5. Let Q denote the parallelogram with vertices s, q, t, p, and define the region $R := Q \cup \nabla_s^p \cup \nabla_s^q$.*

Let y be the vertex such that sy is the edge in C_s^3 and consider the quantity $y_1 := 1 - \|\nabla_s^y\|$ describing the hexagonal distance $\|\cdot\|_{\hexagon}$ from y to the horizontal line through t. In addition to proving that R is empty, the following lemma proves that y_0 and y_1 cannot both be large. Geometrically, s must be near R_t^0 and y must be near R_t^1. See Figure 5.

▶ **Lemma 7.** *If $d_G(s,t) > 5\|st\|_{\bigcirc}$, then int$(R)$ does not contain any vertices of P. Furthermore, $y \in C_t^1$ and*

$$2.5y_0 + 3.5y_1 < 1 \tag{2}$$

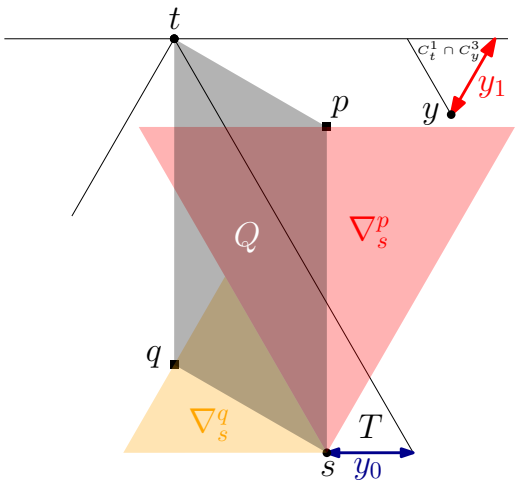


Figure 5 The red triangle is ∇_s^p, the orange triangle is ∇_s^q, and the shaded parallelogram is Q. Their union is R, whose interior can be assumed to not contain any vertices by Lemma 7. Furthermore, there exists a vertex $y \in C_t^1$ close to the horizontal line through t. The triangle T has side-length y_0 and the triangle $C_t^1 \cap C_y^3$ has side-length y_1.

Proof. See Figure 6. If there exists a vertex $w \in Q$ with $w \notin \{s, t\}$, then $\|wt\|_{\hexagon} < 1$. We obtain

$$\begin{aligned} d_G(s,t) &\leq d_G(s,w) + d_G(w,t) && \text{by triangle inequality,} \\ &\leq 5\|sw\|_{\hexagon} + 5\|wt\|_{\hexagon} && \text{by induction,} \\ &= 5\|st\|_{\hexagon} && \text{by } \|\cdot\|_{\hexagon} \text{ collinearity.} \end{aligned}$$

Therefore we may assume parallelogram Q contains no vertices. Next we will rule out the

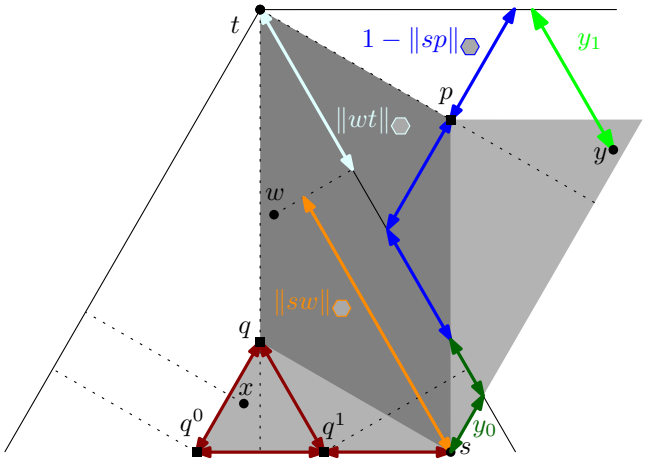


Figure 6 If there is a vertex w in the dark shaded parallelogram, denoted Q, then induction yields paths from s to w and from w to t. If there is an edge sy in the light shaded triangle $\nabla_s^p \setminus Q$, then induction yields a path from y to t. Similarly, if there is an edge sx in the light shaded triangle $\nabla_s^q \setminus Q$, then induction yields a path from x to t.

possibility of $y \in C_t^5 \cup C_t^0$. If $y \in C_t^5$, then we have

$$\begin{aligned} d_G(s,t) &\leq \|sy\|_{\hexagon} + d_G(y,t) && \text{by triangle inequality,} \\ &\leq 1 + 5y_0 && \text{by induction as } \|yt\|_{\bigcirc} \leq \|yt\|_{\hexagon} = \|\nabla_t^y\| \leq \|T\| \\ &< 5\|st\|_{\hexagon} && \text{since } y_0 \leq 0.5 \text{ and } 5 - 2.5y_0 = 5\|st\|_{\hexagon}. \end{aligned}$$

On the other hand if $y \in C_t^0$ or $y \in C_t^1$ below B_t^1, then we use Lemma 5 to obtain $\|sy\|_{\hexagon} \leq 2(\|st\|_{\bigcirc} - \|yt\|_{\bigcirc})$. Indeed, if sy crosses B_t^0, then let $z := sy \cap B_t^0$ and apply Lemma 5 to both sz and zy. We obtain

$$\begin{aligned} d_G(s,t) &\leq \|sy\|_{\hexagon} + d_G(y,t) && \text{by triangle inequality,} \\ &\leq 2(\|st\|_{\bigcirc} - \|yt\|_{\bigcirc}) + 5\|yt\|_{\bigcirc} && \text{by Lemma 5 and induction,} \\ &= 2\|st\|_{\bigcirc} + 3\|yt\|_{\bigcirc} \\ &< 5\|st\|_{\bigcirc} && \text{since } \|yt\|_{\bigcirc} < \|st\|_{\bigcirc}. \end{aligned}$$

Therefore we may assume $y \in C_t^1$ above the bisector B_t^1. We have $\|yt\|_{\hexagon} \leq \|C_t^1 \cap C_s^3\| = 1 - y_0$. Recall that $y_1 := 1 - \|\nabla_s^y\|$, and notice that we also have $y_1 = \|C_t^1 \cap C_y^3\|$. By Lemma 4, $\|yt\|_{\bigcirc} = \|yt\|_{\hexagon} - \frac{1}{2}y_1$. This gives us

$$\begin{aligned} 5 - 2.5y_0 &< d_G(s,t) && \text{by assumption,} \\ &\leq \|sy\|_{\hexagon} + d_G(y,t) && \text{by triangle inequality,} \\ &\leq (1 - y_1) + (5(1 - y_0) - 2.5y_1) && \text{by } \|\nabla_s^y\| = \|sy\|_{\hexagon} \text{ and induction.} \end{aligned} \tag{3}$$

By re-arranging inequality (3), we obtain $2.5y_0 + 3.5y_1 < 1$, proving (2). Notice that $2y_0 + 3(1 - \|sp\|_{\hexagon}) = 1$ by symmetry of isosceles triangles. We have

$$y_1 < \frac{1 - 2.5y_0}{3.5} < \frac{1 - 2y_0}{3} = 1 - \|sp\|_{\hexagon},$$

which implies that the region $\text{int}(\nabla_s^p)$ contains no vertices. Indeed, if there were a point $u \in \text{int}(\nabla_s^p)$, then $1 - y_1 = \|sy\|_{\hexagon} \leq \|su\|_{\hexagon} \leq \|sp\|_{\hexagon}$. Thus far we have shown that Q and ∇_s^p do not contain vertices in their interior, hence it remains to argue that $\text{int}(\nabla_s^q)$ is also empty. Towards a contradiction, assume there is a vertex $x \in \text{int}(\nabla_s^q)$. If x is to the right of the bisector B_t^0, then we have

$$\begin{aligned} d_G(s,t) &\leq \|sx\|_{\hexagon} + d_G(x,t) && \text{by triangle inequality,} \\ &\leq 2(\|st\|_{\bigcirc} - \|xt\|_{\bigcirc}) + 5\|xt\|_{\bigcirc} && \text{by Lemma 5 and induction,} \\ &= 2\|st\|_{\bigcirc} + 3\|xt\|_{\bigcirc} \\ &< 5\|st\|_{\bigcirc} && \text{since } \|xt\|_{\bigcirc} < \|st\|_{\bigcirc}. \end{aligned}$$

On the other hand if x is to the left of the bisector B_t^0, then let $q^0 := R_q^5 \cap R_s^4$ and $q^1 := R_q^0 \cap R_s^4$. Notice that $\|sq^0\|_{\hexagon} = 2\|sq^1\|_{\hexagon}$. We have

$$\begin{aligned} d_G(s,t) &\leq \|sx\|_{\hexagon} + d_G(x,t) && \text{by triangle inequality,} \\ &\leq \|sq^0\|_{\hexagon} + 5\|xt\|_{\bigcirc} && \text{by induction,} \\ &\leq 2\|sq^1\|_{\hexagon} + 5\|q^0t\|_{\bigcirc} && \text{since } \|xt\|_{\bigcirc} \leq \|q^0t\|_{\bigcirc}, \\ &= 2(2(\|st\|_{\bigcirc} - \|q^1t\|_{\bigcirc})) + 5\|q^1t\|_{\bigcirc} && \text{by Lemma 5 and } \|q^0t\|_{\bigcirc} = \|q^1t\|_{\bigcirc}, \\ &= 4\|st\|_{\bigcirc} + \|q^1t\|_{\bigcirc} \\ &\leq 5\|st\|_{\bigcirc} && \text{since } \|q^1t\|_{\bigcirc} \leq \|st\|_{\bigcirc}. \end{aligned}$$

Therefore the region $\text{int}(\nabla_s^q)$ contains no vertices. This completes the proof. ◀

Now that we have established an empty region R and the position of y, we would like to analyse the greedy path from y to t. Recall that for any vertices $u, v \in P$, the greedy path from u to v is denoted $\pi(u, v)$. Every edge ab of $\pi(u, v)$ satisfies $b \in C_a^i \iff v \in C_a^i$. Intuitively, the greedy path simply follows the edge in the direction of the destination. In [1] the authors show that for every edge uv of $\pi(s, t)$, we have $\|ut\|_{\hexagon} > \|vt\|_{\hexagon}$. However, this does not guarantee that the greedy path is an efficient path to t. The main reason is that $\pi(s, t)$ may spiral around t arbitrarily many times while making negligible progress. The authors of [1] show how to avoid a spiraling greedy path by assuming there is an edge from t to s and using the empty triangle ∇_t^s as an obstruction. This is done in Lemmas 2, 3 and 4 from [1], which we restate here using our notation.

▶ **Lemma 8** (Lemma 2 in [1]). *If edge uv is on $\pi(u, t)$ and $u \in C_t^i$, then $v \in C_t^{i-1} \cup C_t^i \cup C_t^{i+1}$.*

▶ **Lemma 9** (Lemma 3 in [1]). *Let $s, t \in P$ and assume ts is an edge of $\vec{\Theta}_6(P)$. Assume the vertices a, b, c appear in this order in $\pi(s, t)$, not necessarily distinct or consecutive. Then if a and c are both in the same cone C_t^i for $i \in \mathbb{Z}_6$, then $int(\nabla_a^b) \cap int(\nabla_t^c) = \emptyset$.*

▶ **Lemma 10** (Lemma 4 in [1]). *Let $s, t \in P$ and assume ts is an edge of $\vec{\Theta}_6(P)$. Then for $i \in \mathbb{Z}_6$, we have*

$$\sum_{\substack{ab \in \pi(s,t) \\ a \in C_t^i}} \|ab\| \leq \|\nabla_t^s\|$$

Instead of working with an empty triangle that forces the greedy path not to spiral as in [1], we will ignore any spiraling by only using the greedy path up to and including the first edge that crosses the boundary R_t^0. Despite this change, the arguments in [1] can be applied in our setting to prove Lemma 11 which is a slightly modified version of Lemma 4 from [1].

▶ **Lemma 11.** *Fix $j \in \mathbb{Z}_6$ and let $s, t, u, v \in P$ such that uv is an edge of $\pi(s, t)$ and no edge of the truncated greedy path $\pi(s, t)_u$ crosses R_t^j. For $i \in \mathbb{Z}_6$, let f_i denote the first vertex of $\pi(s, t)_u$ in C_t^i. If no such vertex f_i exists, let $f_i := t$. Then $\|\pi(s, t)_v\|_{\hexagon} \leq \sum_{i=0}^{5} \|f_i t\|_{\hexagon}$.*

Proof. Suppose without loss of generality that $i = 0$. For any point p below t, let p' denote the intersection of the horizontal line through p with the ray R_t^0. We claim that if ab and cd are edges of $\pi(s, t)_v$ such that $a, c \in C_t^0$, then the segments $a'b'$ and $c'd'$ are disjoint. Furthermore, both $a'b'$ and $c'd'$ are subsegments of $f_0't$. This claim follows from Lemma 9 of [1] since the path $\pi(s, t)_u$ is assumed to not cross R_t^j. We sum all edges with source in C_t^0 to obtain

$$\sum_{\substack{ab \text{ on } \pi(s,t)_v \\ a \in C_t^0}} \|ab\|_{\hexagon} = \sum_{\substack{ab \text{ on } \pi(s,t)_v \\ a \in C_t^0}} \|a'b'\|_{\hexagon} \leq \|f_0't\|_{\hexagon} = \|f_0 t\|_{\hexagon}.$$

The lemma follows by summing over the edges in all cones C_t^i for $i \in \mathbb{Z}_6$. ◀

For the rest of the section, we make Assumptions 12 and 13 to help us prove Theorem 3. For the cases when these assumptions do not hold, see the full version [5]. The first assumption roughly states that the greedy path from y spirals counterclockwise around t without making significant progress to t. We define $u^y v^y$ to be the first edge of $\pi(y, t)$ which crosses R_t^0. If such an edge does not exist, let $u^y := t$ and $v^y := t$. Then the first assumption can be stated as follows:

▶ **Assumption 12.** *There exists a vertex $q \in C_t^5$ on $\pi(y, t)_{v^y}$ with $\|qt\|_{\hexagon} > y_0$.*

Our second assumption is that there is no edge uv crossing R_t^5 such that $u \in T$ and $v \in C_u^4$.

▶ **Assumption 13.** *For every vertex $u \in T$, we have $P \cap int(C_t^0 \cap C_u^4) \neq \emptyset$.*

Let sx be the edge in C_s^4. By Assumption 13, x exists and $x \in C_t^0$. Combining our two assumptions allows us to naturally consider two candidate paths, loosely referred to as the *x-path* and the *y-path*, and show that one of them is sufficiently short to force a contradiction with $d_G(s,t) > 5\|st\|_{\bigcirc}$. Informally, the x-path travels clockwise around t, whereas the y-path travels in the opposite direction. Where these two paths meet, they must compete for space. The lengths of the paths are primarily dictated by their *long* edges which cross cone boundaries around t. These *long* edges come with large empty regions which squeeze the other path closer to t. We formalize this intuition below.

Recall that $u^y v^y$ is the first edge of $\pi(y,t)$ that crosses the boundary R_t^0. Consider the candidate path sy concatenated with $\pi(y,t)_{v^y}$, which we loosely refer to as the y-path, ending at v^y. Overall, the y-path progresses counterclockwise about t until C_t^5, however it is still possible that the y-path reverses direction and ends up crossing R_t^0 from C_t^1 to C_t^0. Maintaining full generality, we may only assume that for some $i \in \{0,1\}$, we have $u^y \in C_t^i$ and $v^y \in C_t^{1-i}$, which follows from Lemma 8. In light of Lemma 11, we need not concern ourselves with every edge of this path. An upper bound can be established from only the first vertex of each cone. In fact, we only need to upper bound the value $\|f_i t\|_{\hexagon}$ where f_i is the first vertex of C_t^i, hence we look to the previous cone for structure. For $i \in \{1,2,3,4,5\}$, let $y^i \in C_t^i$ be the vertex of $\pi(y,t)_{u^y}$ which maximizes $\|C_t^i \cap C_{y^i}^{i-2}\|$. See Figure 7 for example.

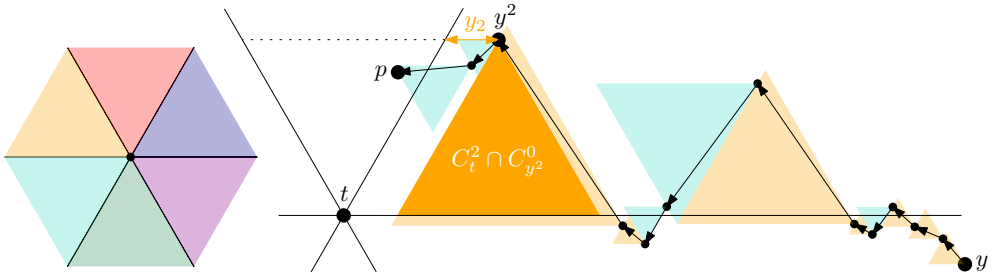


Figure 7 In this example, p is the first vertex of $\pi(y,t)_{v^y}$ in cone C_t^3. The highest vertex of $\pi(y,t)_{v^y}$ in C_t^2 is y^2 by definition. The maximal triangle $C_t^2 \cap C_{y^2}^0$ is coloured orange, and $y_2 = \|y^2 t\|_{\hexagon} - \|C_t^2 \cap C_{y^2}^0\|$. Observation 14 implies that y^2 has a greater y-coordinate than p, thus $\|C_t^2 \cap C_{y^2}^0\|$ provides an upper bound on $\|pt\|_{\hexagon}$.

▶ **Observation 14.** *For $i \in \{1,2,3,4,5\}$, no vertex of $\pi(y,t)_{y^i}$ is in $int(C_t^{i+1})$.*

Proof. The observation holds for $i = 1$ since $y^1 = y$. Then for $i \in \{2,3,4,5\}$, consider the first vertex v of C_t^{i+1} on $\pi(y,t)_{u^y}$. Then y^i and t must be on opposite sides of the line $R_v^{i-1} \cup R_v^{i+2}$, which implies $\|y^i t\|_{\hexagon} > \|vt\|_{\hexagon}$, meaning that y^i appears before v on $\pi(y,t)_{u^y}$. ◀

Observation 14 implies y^1, y^2, y^3, y^4, y^5 appear in order on $\pi(y,t)_{u^y}$. Intuitively, y^i is the head of a *long* edge whose tail is in the previous cone C_t^{i-1}. Indeed, if the tail was in the same cone as y^i, then $\|C_t^i \cap C_{y^i}^{i-2}\|$ would not be maximal. We define the distance y_i from y^i to C_t^{i+1} as $y_i := \|C_t^i \cap C_{y^i}^{i+2}\|$. Equivalently, $y_i = \|y^i t\|_{\hexagon} - \|C_t^i \cap C_{y^i}^{i-2}\|$. By Assumption 12, the y-path must pass through cones $C_t^1, \ldots, C_t^5$, hence y_i is well-defined for $i \in \{1,2,3,4,5\}$. Also notice that the region $\text{int}(C_t^i \cap C_{y^i}^{i-2})$ is empty since y^i is necessarily the head of an

edge from C_t^{i-1}. Next, for each $i \in \mathbb{Z}_6$, we let Y_i be the value of $\|ut\|_{\hexagon}$ where u is the first vertex of $\pi(y,t)_{u^y}$ with $u \in C_t^i$. If no such vertex $u \in C_t^i$ exists, then $Y_i := 0$. First, we have $Y_1 = \|yt\|_{\hexagon} \le 1 - y_0$. Next, for $i \in \{1,2,3,4,5\}$, cone C_t^{i+1} can only be visited from cone C_t^i on $\pi(y,t)_{u^y}$, therefore we see that for $i \in \{1,2,3,4,5\}$,

$$y_0 + Y_1 \le 1 \qquad \text{and} \qquad y_i + Y_{i+1} \le y_i + \|C_t^i \cap C_{y^i}^{i-2}\| = \|y^i t\|_{\hexagon} \le Y_i \tag{4}$$

We include these inequalities in our system which will eventually be proven infeasible. Next we will discuss the second candidate path: the x-path. Recall sx is the edge in $C_s^4 \cap C_t^0$. We will define the analogous variables for x as we did for y. Let $u^x v^x$ be the first edge of $\pi(x,t)$ that crosses R_t^0. As before, Lemma 8 implies that for some $i \in \{0,1\}$, we have $u^x \in C_t^i$ and $v^x \in C_t^{1-i}$. If no such crossing exists, then set $u^x := t$ and $v^x := t$. For $i \in \{0,5,4,3,2\}$, let $x^i \in C_t^i$ be the vertex of $\pi(x,t)_{u^x}$ which maximizes $\|C_t^i \cap C_{x^i}^{i+2}\|$. We define the distance x_i from x^i to C_t^{i-1} as $x_i := \|C_t^i \cap C_{x^i}^{i-2}\|$. If no such x^i exists, then define $x_i := 0$. However, we will later argue in Claim 19 that the vertices x^0, x^5, x^4, x^3 must exist, otherwise the x-path is sufficiently short to force a contradiction. Observe that for $i \in \{5,4,3,2\}$ the region $C_t^i \cap C_{x^i}^{i+2}$ is empty since x^i is the head of an edge from C_t^{i+1}. Next, for each $i \in \mathbb{Z}_6$, we let X_i be the value of $\|ut\|_{\hexagon}$ where u is the first vertex of $\pi(x,t)_{u^x}$ with $u \in C_t^i$. If no such vertex u exists, then $X_i := 0$. Since greedy paths decrease in $\|\cdot\|_{\hexagon}$-norm, then we have $X_0 = \|xt\|_{\hexagon} \le 1$. Next, for $i \in \{5,4,3,2,1\}$, if v is the first vertex of $\pi(x,t)_{u^x}$ in C_t^i, then its predecessor u must be in cone C_t^{i+1}, therefore we see that for $i \in \{5,4,3,2,1\}$,

$$X_0 \le 1 \quad \text{and} \quad x_{i+1} + X_i \le x_{i+1} + \|C_t^{i+1} \cap C_{x^{i+1}}^{i+3}\| = \|x^{i+1}t\|_{\hexagon} \le X_{i+1}. \tag{5}$$

Intuitively, the only vertices in cone C_t^i that are important to us are x^i and y^i. Now we consider possible paths from s to t. We could take the y-path from s to v^y, then apply induction at v^y. By Lemma 11, this yields

$$\begin{aligned} 5 - 2.5y_0 &\le d_G(s,t) \\ &\le \|sy\|_{\hexagon} + \|\pi(y,t)_{v^y}\|_{\hexagon} + d_G(v^y,t) \\ &\le (1-y_1) + (Y_1+Y_2+Y_3+Y_4+Y_5+Y_0) + 5||v^y t||_{\bigcirc}. \end{aligned} \tag{6}$$

On the other hand, we could instead follow the x-path from s to v^x, then apply induction at v^x. Similarly, we obtain

$$\begin{aligned} 5 - 2.5y_0 &\le d_G(s,t) \\ &\le \|sx\|_{\hexagon} + \|\pi(x,t)_{v^x}\|_{\hexagon} + d_G(v^x,t) \\ &\le (1-y_0-x_0) + (X_0+X_5+X_4+X_3+X_2+X_1) + 5||v^x t||_{\bigcirc}. \end{aligned} \tag{7}$$

While we have proven several inequalities (2),(4),(5),(6),(7), they are not yet strong enough to form an infeasible system. First, notice that since R may not contain vertices, then $u^y v^y$ must cross both B_t^0 and B_t^1. By the following lemma, this implies that the edge $u^y v^y$ makes significant progress towards t.

▶ **Lemma 15.** *Let $u, v, t \in P$ with edge uv contained in ∇_u^t. Assume uv crosses both B_t^0 and B_t^1, then $2\|vt\|_{\bigcirc} \le \|uv\|_{\hexagon}$.*

Proof. Suppose without loss of generality that $u \in C_t^0$ with u to the left of B_t^0, and $v \in C_t^1$ with v above B_t^1. See Figure 8. Consider the point $t' := B_t^0 \cap R_v^5$. Since u is to the left of B_t^0, then the segment vt' is contained in ∇_u^v, hence $\|vt'\|_2 \le \|\nabla_u^v\| = \|uv\|_{\hexagon}$. On the other hand, consider the point $v' := R_t^1 \cap B_v^3$. Then considering that vt' and $v't$ have respective slopes $\sqrt{3}$ and 0, then we have $2\|vt\|_{\bigcirc} = 2\|v't\|_2 = \|vt'\|_2 \le \|uv\|_{\hexagon}$. ◀

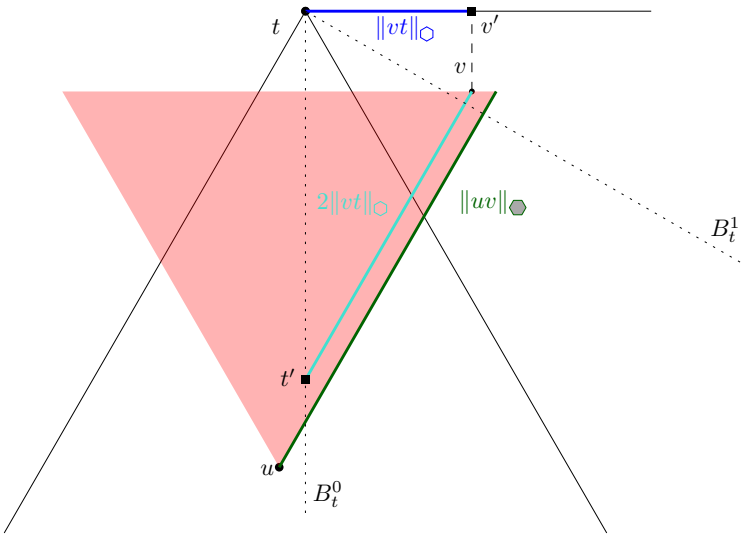


Figure 8 Lemma 15 where $u \in C_t^0$ with u to the left of B_t^0, and $v \in C_t^1$ with v above B_t^1.

Lemma 15 quantifies the toll required for an edge to cross the empty region. Intuitively, the distance to t is halved each time the empty region R is crossed. In order to also apply Lemma 15 to $u^x v^x$, we require two observations, which follow from Assumptions 12 and 13.

▶ **Observation 16.** *If x^1 exists, then $\|x^1 t\|_{\hexagon} \leq 1 - y_0$.*

Proof. Suppose on the contrary that $\|x^1 t\|_{\hexagon} > 1 - y_0$. See Figure 9. Since greedy paths decrease in $\|\cdot\|_{\hexagon}$-norm to their destination, then $\|y^i t\|_{\hexagon}$ is monotonically decreasing, whereas $\|x^i t\|_{\hexagon}$ is monotonically increasing for $i \in \{1, 2, 3, 4, 5\}$. This means that $\|y^i t\|_{\hexagon} \leq \|y^1 t\|_{\hexagon} \leq 1 - y_0 < \|x^1 t\|_{\hexagon} \leq \|x^i t\|_{\hexagon}$. In particular, the empty regions $C_t^i \cap C_{x^i}^{i+2}$ for $i \in \{2, 3, 4\}$ imply that $Y_3 \leq x_2$, $Y_4 \leq x_3$, and $Y_5 \leq x_4$. Assumption 12 implies $Y_5 \geq y_0$, therefore (4) implies $\min(x_2, x_3, x_4) \geq y_0$. However this yields $1 - y_0 < \|x^1 t\|_{\hexagon} < 1 - x_0 - x_5 - x_4 - x_3 - x_2 \leq 1 - 3y_0$, which implies $y_0 < 0$. This is a contradiction, hence we may assume $\|x^1 t\|_{\hexagon} \leq 1 - y_0$ for the rest of the proof. ◀

▶ **Observation 17.** *There is no edge uv on $\pi(x, t)_{v^x}$ with $u \in C_t^5$ and $v \in T$.*

Proof. If there were such an edge uv on $\pi(x, t)_{v^x}$ with $u \in C_t^5$ and $v \in T$, the empty region $\text{int}(\nabla_u^v)$ contradicts Assumption 13. See Figure 10. ◀

Observations 16 and 17 together imply that the edge $u^x v^x$ crosses both B_t^0 and B_t^1 since the interior of region R is empty by Lemma 7. Therefore Lemma 15 applies to $u^x v^x$ as well. Furthermore, we can ensure that $\|v^x t\|_{\hexagon} \leq x_0$ by the following claim.

▷ Claim 18. There is no vertex to the left of B_t^0 in the interior of C_x^2.

Proof. Assume there is such a vertex $p \in C_x^2$ to the left of B_t^0. Note that we must have $p \in C_s^3$ since $C_x^2 \cap C_s^4 \subseteq \nabla_s^x$ and ∇_s^x contains no vertices. This means that p is above ∇_s^y, yielding $\|pt\|_{\hexagon} \leq y_1$. See Figure 11. Let u^1 be the vertex of $T := C_t^0 \cap C_s^2$ minimizing $\|u^1 t\|_{\hexagon}$. Then we take the edge $u^1 u^2$ in $C_{u^1}^4$. By Assumption 13, we have $u^2 \in C_t^0$. If $u^2 \in C_x^2$, then set $u^3 := u^2$. Otherwise take the edge $u^2 u^3$ in $C_{u^2}^2$. We must have $u^3 \in C_s^3$, hence we apply induction from u^3 to t. This yields

$$\begin{aligned} 5 - 2.5y_0 &< d_G(s, t) \\ &\leq d_G(s, u^1) + \|u^1 u^2\|_{\hexagon} + \|u^2 u^3\|_{\hexagon} + d_G(u^3, t) \\ &\leq 5y_0 + 1 + 1 + 5y_1. \end{aligned} \tag{8}$$

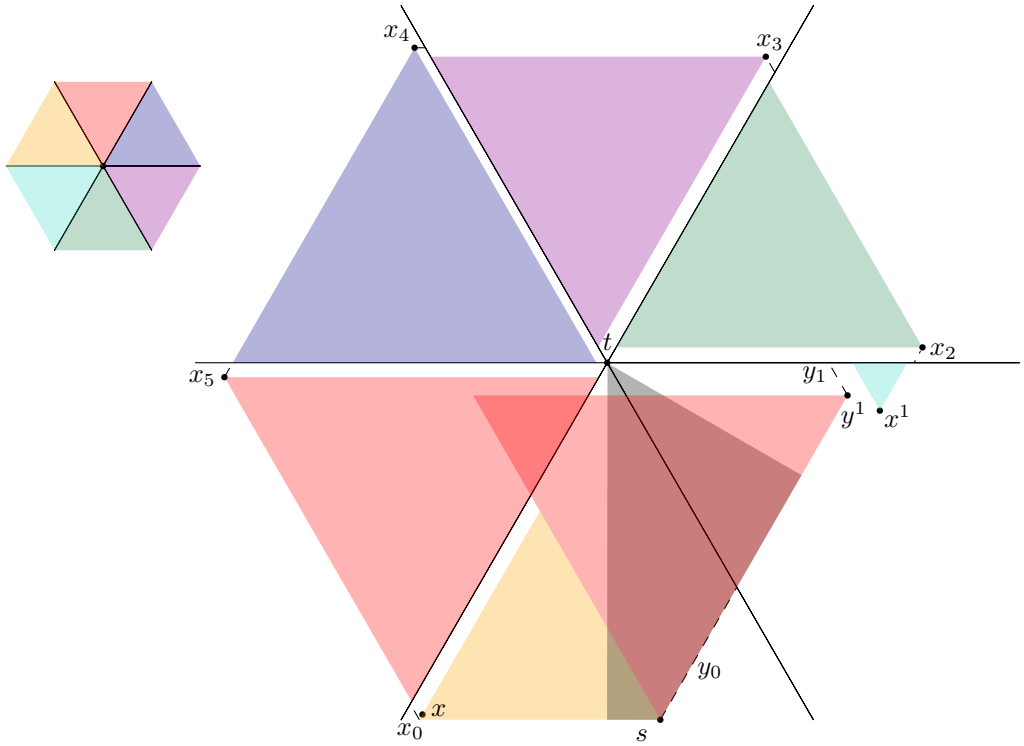


Figure 9 Observation 16: From Assumption 12, there cannot exist x^1 such that $\|x^1 t\| > 1 - y_0$.

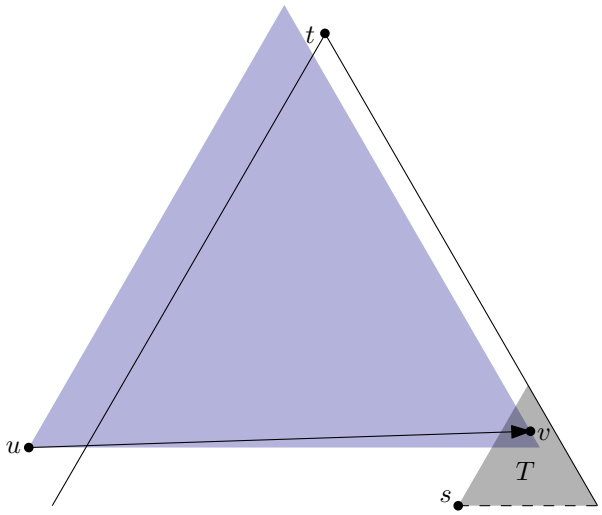


Figure 10 Observation 17: If uv is an edge from C_t^5 to T, then the region $C_v^4 \cap C_t^0$ contains no vertices in its interior, contradicting Assumption 13.

However, the system of inequalities (2,8) is infeasible. Therefore we may assume that the region $C_x^2 \cap C_t^0$ to the left of B_t^0 is empty for the remainder of the proof. ◁

In order to further restrict the set of solutions to our linear system of inequalities towards infeasibility, we will consider several cases with more granularity. We will first consider the direction in which the edge $u^y v^y$ crosses the boundary R_t^0 in order to strengthen our upper

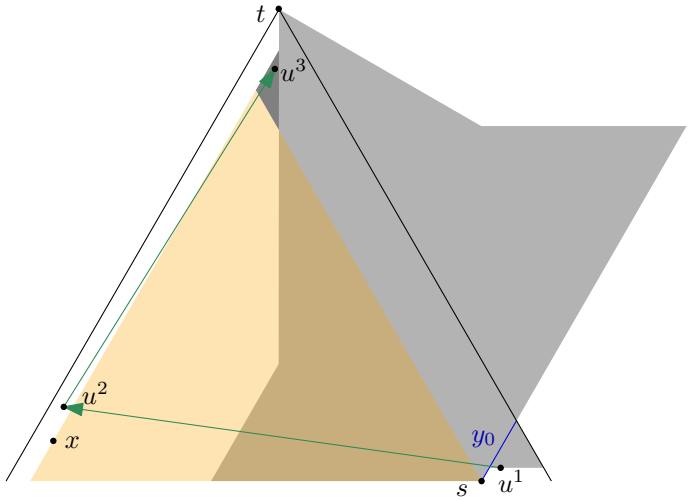


Figure 11 If there is a vertex to the left of B_t^0 in C_x^2 (the dark grey region), then we can form a path that passes through s, u^1, u^2, u^3, t. Induction is used to bound $d_G(s, u^1)$ and $d_G(u^3, t)$, whereas u^1u^2 and u^2u^3 are edges in G (in green).

bound on $\|v^y t\|_{\bigcirc}$. Recall that $5\|v^y t\|_{\bigcirc}$ accounts for the path from v^y to t by induction, hence it is an important quantity to reduce. We will apply the same analysis to $u^x v^x$ to also upper bound $\|v^x t\|_{\bigcirc}$, then lastly we will consider the trade-off between the x-path and y-path. In Figure 12, we show an example of the x-path and y-path that compete for space.

Next, we show in Claim 19 that the x-path must visit the cone C_t^3.

▷ Claim 19. The path $\pi(x,t)_{v^x}$ must contain a vertex in C_t^3.

Proof. If the claim did not hold, then $u^x v^x$ would necessarily cross R in the counterclockwise direction. See Figure 13. If $\pi(x,t)_{v^x} \cap C_t^5 \neq \emptyset$, then we use Lemma 11 and apply induction at v^x to obtain

$$\begin{aligned} 5 - 2.5y_0 &< d_G(s,t) \\ &\leq \|sx\|_{\hexagon} + \|\pi(x,t)_{v^x}\|_{\hexagon} + d_G(v^x,t) \\ &\leq (1 - x_0 - y_0) + (X_0 + X_5 + X_4) + 5||v^x t||_{\hexagon} \\ &\leq (1 - x_0 - y_0) + 1 + (1 - x_0) + (1 - x_0 - x_5) + 5\min(x_0, \frac{x_5}{2}). \end{aligned} \tag{9}$$

However, the linear system (2,9) is infeasible. On the other hand if $\pi(x,t)_{v^x} \cap C_t^5 = \emptyset$, then

$$\begin{aligned} d_G(s,t) &\leq \|sx\|_{\hexagon} + \|\pi(x,t)_{v^x}\|_{\hexagon} + d_G(v^x,t) \\ &\leq (1 - x_0 - y_0) + X_0 + 5||v^x t||_{\hexagon} \\ &\leq (1 - x_0 - y_0) + 1 + 5x_0 \\ &\leq 4 - y_0 && \text{since } x_0 \leq 1/2, \\ &\leq 5 - 2.5y_0 && \text{since } y_0 \leq 1/2. \end{aligned}$$

which is a contradiction. To summarize, $\pi(x,t)_{v^x}$ must enter C_t^3. ◁

First, notice that $u^y v^y$ can cross R_t^0 either in the clockwise or counterclockwise direction:

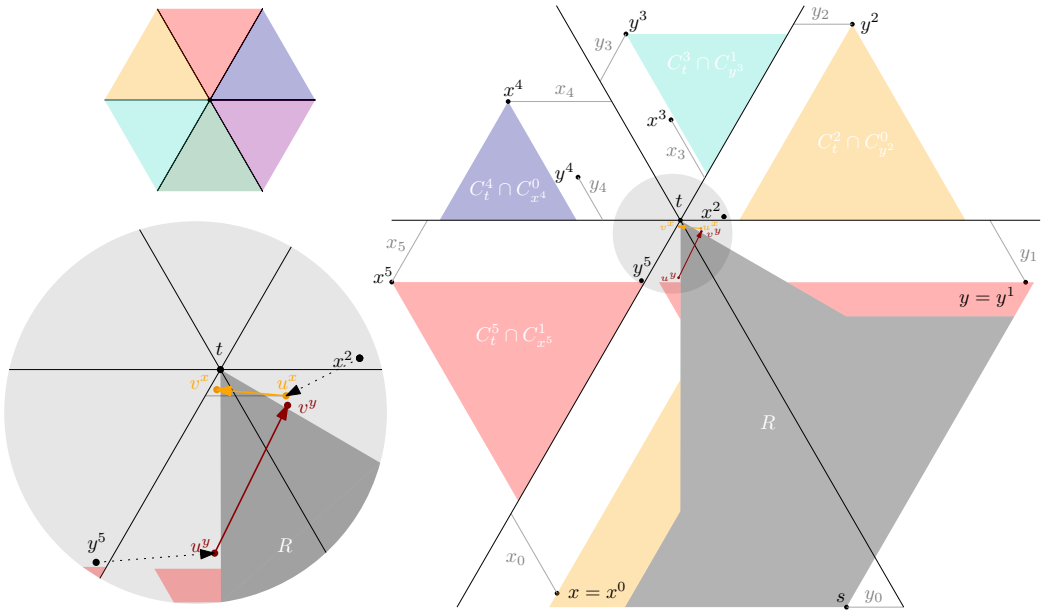


Figure 12 Top Left: The colour coding for edges based on direction. Right: The grey region R contains no vertices in its interior by Lemma 7. The triangular regions are coloured according to their edge direction. This example shows the case when $v^y \in C_t^1$ (case $Y1$), $v^x \in C_t^0$ (case $X0$), and $j = 3$ (case J3). Notice that the empty turquoise region $C_t^3 \cap C_{y^3}^1$ constrains the position of x^3, forcing x^2 to be considerably closer to t. Lower Left: This enlarged view shows the edges that cross the empty grey region R. Notice that the edges $u^x v^x$ and $u^y v^y$ both make considerable progress towards t (quantified in Lemma 15).

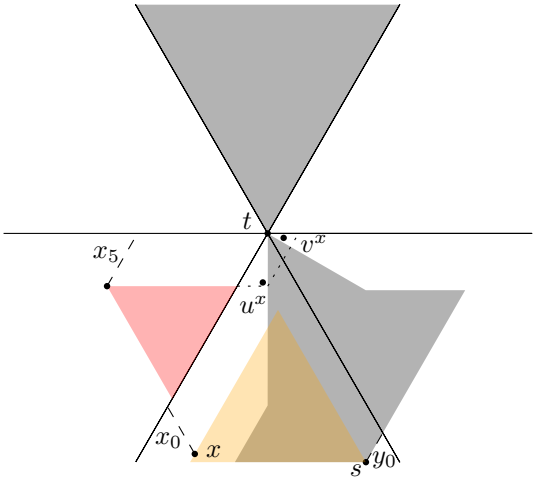


Figure 13 Claim 19: If $\pi(x,t)_{v^x}$ does not enter C_t^3, then we can form an infeasible system of linear inequalities.

Case Y0: $v^y \in C_t^0$. Then we must have $u^y \in C_t^1$. We obtain the following two inequalities:

$$||v^y t||_{\bigcirc} \leq y_1, \qquad \text{by empty region int}(\nabla_s^y). \tag{10}$$

$$||v^y t||_{\bigcirc} \leq \frac{1}{2}\min(y_2, y_3, y_4), \qquad \text{by Assumption 12 and Lemma 15.} \tag{11}$$

Case Y1: $v^y \in C_t^1$. Then we must have $u^y \in C_t^0$. We obtain

$$||v^y t||_{\bigcirc} \leq x_0, \qquad \text{by Claim 18.} \tag{12}$$

$$||v^y t||_{\bigcirc} \leq \frac{Y_0}{2}, \qquad \text{by Lemma 15.} \tag{13}$$

Similarly, $u^x v^x$ can also cross R_t^0 in one of two possible ways.

Case X0: $v^x \in C_t^0$. Then we have

$$||v^x t||_{\bigcirc} \leq y_1, \qquad \text{by Observation 16 and empty region int}(\nabla_s^y). \tag{14}$$

$$||v^x t||_{\bigcirc} \leq \frac{X_1}{2}, \qquad \text{by Lemma 15.} \tag{15}$$

Case X1: $v^x \in C_t^1$. Then $u^y \in C_t^0$. We have

$$||v^x t||_{\bigcirc} \leq x_0, \qquad \text{by Claim 18 and 17.} \tag{16}$$

$$||v^x t||_{\bigcirc} \leq \frac{1}{2}\min(x_5, x_4, x_3), \qquad \text{by Claim 19 and Lemma 15.} \tag{17}$$

Next, we will derive several useful inequalities that reflect the balance between the two candidate paths. Intuitively, if $\pi(y,t)_{v^y}$ is long, then the empty regions $C_t^i \cap C_{y^i}^{i-2}$ must be large, leaving no room for the other path $\pi(x,t)_{v^x}$. We formalize this trade-off as follows. Let j be the greatest index in $\{2,3,4,5\}$ such that $\|C_t^j \cap C_{y^j}^{j-2}\| > x_j$. For example when $j=3$, see Figure 12.

Case: No such j exists. Then by definition we must have

$$Y_{i+1} \leq \|C_t^i \cap C_{y^i}^{i-2}\| \leq x_i \quad \text{for } 2 \leq i \leq 5. \tag{18}$$

Case: $j \in \{2,3,4,5\}$. As j is maximal, then $\|C_t^i \cap C_{y^i}^{i-2}\| \leq x_i$ for $j < i \leq 5$. Next, since $\|C_t^j \cap C_{y^j}^{j-2}\| > x_j$, then we must have $\|y^k t\|_{\hexagon} > \|x^k t\|_{\hexagon}$ for $2 \leq k \leq j$. By the empty regions $C_t^i \cap C_{y^i}^{i-2}$, we must have

$$Y_{i+1} \leq \|C_t^i \cap C_{y^i}^{i-2}\| \leq x_i \quad \text{for } j < i \leq 5, \quad \text{and} \quad X_{k-1} \leq y_k \quad \text{for } 2 \leq k \leq j. \tag{19}$$

We denote the cases where $j \in \{2,3,4,5\}$ as J2, J3, J4, J5, and observe that for each case, we yield four inequalities from (19).

Under Assumptions 12 and 13 we have the following possibility space: {Y0, Y1} × {X0, X1} × {(18), J2, J3, J4, J5}. In total there are $2 \times 2 \times 5 = 20$ possibilities, and in every case, the inequalities (2), (4), (5), (6), (7) also hold. It can be shown that the linear system of inequalities is infeasible for 16 of the 20 possibilities. In the full version [5], we show that for the remaining four cases, we can define a third candidate path which is sufficiently short for a contradiction. These remaining four cases are (X1, Y1, J2), (X1, Y1, J3), (X1, Y1, J4), (X0, Y0, J3).

References

1 Hugo A. Akitaya, Ahmad Biniaz, and Prosenjit Bose. On the spanning and routing ratios of the directed Θ_6-graph. *Comput. Geom.*, 105-106:101881, 2022. doi:10.1016/j.comgeo.2022.101881.

2 Luis Barba, Prosenjit Bose, Jean-Lou De Carufel, André van Renssen, and Sander Verdonschot. On the stretch factor of the theta-4 graph. In *WADS*, volume 8037 of *Lecture Notes in Computer Science*, pages 109–120. Springer, 2013. doi:10.1007/978-3-642-40104-6_10.

3 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perkovic. Tight stretch factors for L_1- and L_∞-Delaunay triangulations. *Comput. Geom.*, 48(3):237–250, 2015. doi:10.1016/j.comgeo.2014.10.005.

4 Prosenjit Bose, Jean-Lou De Carufel, Darryl Hill, and Michiel Smid. On the spanning and routing ratio of the directed theta-four graph. *Discret. Comput. Geom.*, 71(3):872–892, 2024. doi:10.1007/s00454-023-00597-8.

5 Prosenjit Bose, Jean-Lou De Carufel, Darryl Hill, and John Stuart. The spanning ratio of the directed θ_6-graph is 5, 2026. arXiv:2603.09048.

6 Prosenjit Bose, Jean-Lou De Carufel, Pat Morin, André van Renssen, and Sander Verdonschot. Towards tight bounds on theta-graphs: More is not always better. *Theor. Comput. Sci.*, 616:70–93, 2016. doi:10.1016/j.tcs.2015.12.017.

7 Prosenjit Bose, Jean-Lou De Carufel, and Sandrine Njoo. The exact spanning ratio of the parallelogram Delaunay graph. *Theor. Comput. Sci.*, 1037:115152, 2025. doi:10.1016/j.tcs.2025.115152.

8 Prosenjit Bose and André van Renssen. Upper bounds on the spanning ratio of constrained theta-graphs. In *LATIN*, volume 8392 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 2014. doi:10.1007/978-3-642-54423-1_10.

9 Paul Chew. There is a planar graph almost as good as the complete graph. In *SCG*, pages 169–177. ACM, 1986. doi:10.1145/10515.10534.

10 Paul Chew. There are planar graphs almost as good as the complete graph. *J. Comput. Syst. Sci.*, 39(2):205–219, 1989. doi:10.1016/0022-0000(89)90044-5.

11 K. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 56–65, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.28402.

12 David P. Dobkin, Steven J. Friedman, and Kenneth J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discret. Comput. Geom.*, 5:399–407, 1990. doi:10.1007/BF02187801.

13 J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discret. Comput. Geom.*, 7(1):13–28, December 1992. doi:10.1007/BF02187821.

14 Anna Lubiw and Debajyoti Mondal. Construction and local routing for angle-monotone graphs. *J. Graph Algorithms Appl.*, 23(2):345–369, 2019. doi:10.7155/jgaa.00494.

15 Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007. doi:10.1017/CBO9780511546884.

16 Ljubomir Perkovic, Michael Dennis, and Duru Türkoglu. The stretch factor of hexagon-Delaunay triangulations. *J. Comput. Geom.*, 12(2):86–125, 2021. doi:10.20382/jocg.v12i2a5.

17 J. Ruppert and R. Seidel. Approximating the D-dimensional complete Euclidean graph. In *Proceedings of the 3rd Annual Canadian Conference on Computational Geometry*, pages 207–210, 1991. URL: https://cccg.ca/proceedings/1991/paper50.pdf.

18 John Stuart, Jean-Lou De Carufel, and Prosenjit Bose. The exact routing and spanning ratio of arbitrary triangle Delaunay graphs. In Rahnuma Islam Nishat, editor, *Proceedings of the 36th Canadian Conference on Computational Geometry, CCCG 2024, Brock University, St. Catharines, Canada, July 17-19, 2024*, pages 83–89, 2024. doi:10.48550/arXiv.2506.12625.

19 André van Renssen, Yuan Sha, Yucheng Sun, and Sampson Wong. The tight spanning ratio of the rectangle Delaunay triangulation. In *ESA*, volume 274 of *LIPIcs*, pages 99:1–99:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.99.

20 Ge Xia. The stretch factor of the Delaunay triangulation is less than 1.998. *SIAM J. Comput.*, 42(4):1620–1659, 2013. doi:10.1137/110832458.

21 Ge Xia and Liang Zhang. Toward the tight bound of the stretch factor of Delaunay triangulations. In *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry, Toronto, Ontario, Canada, August 10-12, 2011*, 2011. URL: http://www.cccg.ca/proceedings/2011/papers/paper57.pdf.

22 Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4):721–736, 1982. doi:10.1137/0211059.

On Minimum Venn Diagrams

Sofia Brenner
Institute of Mathematics, University of Kassel, Germany

Petr Gregor
Department of Theoretical Computer Science and Mathematical Logic, Charles University, Prague, Czech Republic

Torsten Mütze
Institute of Mathematics, University of Kassel, Germany

Francesco Verciani
Institute of Mathematics, University of Kassel, Germany

Abstract

An *n-Venn diagram* is a diagram in the plane consisting of n simple closed curves that intersect only finitely many times such that each of the 2^n possible intersections of their interiors is represented by a single connected region. An n-Venn diagram has at most $2^n - 2$ crossings, and if this maximum number of crossings is attained, then only two curves intersect in every crossing. To complement this, Bultena and Ruskey considered n-Venn diagrams that minimize the number of crossings, which implies that many curves intersect in every crossing. Specifically, they proved that the total number of crossings in any n-Venn diagram is at least $L_n := \lceil \frac{2^n-2}{n-1} \rceil$, and if this lower bound is attained, then essentially all n curves intersect in every crossing. Diagrams achieving this bound are called *minimum* Venn diagrams, and are known only for $n \leq 7$. Bultena and Ruskey conjectured that they exist for all $n \geq 8$. In this work, we establish an asymptotic version of their conjecture. For $n = 8$ we construct a diagram with 40 crossings, only 3 more than the lower bound $L_8 = 37$. Furthermore, for every n of the form $n = 2^k$ for some integer $k \geq 4$, we construct an n-Venn diagram with at most $(1 + \frac{33}{8n})L_n = (1 + o(1))L_n$ many crossings. Via a doubling trick this also gives $(n+m)$-Venn diagrams for all $0 \leq m < n$ with at most $40 \cdot 2^m$ crossings for $n = 8$ and at most $(1 + \frac{33}{8n})\frac{n+m}{n}L_{n+m} = (2 + o(1))L_{n+m}$ many crossings for $k \geq 4$. In particular, we obtain n-Venn diagrams with the smallest known number of crossings for all $n \geq 8$. Our constructions are based on partitions of the hypercube into isometric paths and cycles, using a result of Ramras.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Discrete mathematics

Keywords and phrases Venn diagram, crossing, conjecture, hypercube, partition

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.21

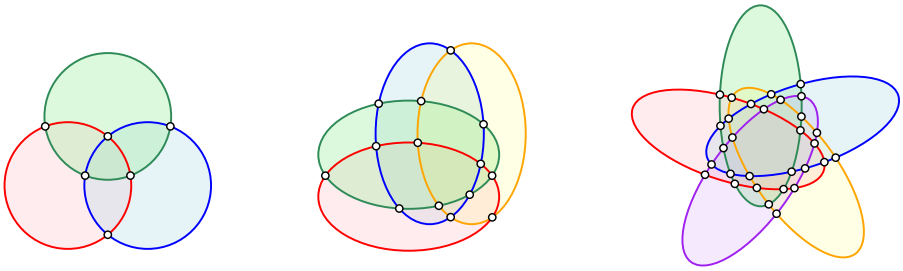
Related Version *Full Version*: `https://arxiv.org/abs/2511.09230` [1]

Funding Sofia Brenner, Torsten Mütze, and Francesco Verciani were supported by German Science Foundation (DFG) grant 522790373. Sofia Brenner also acknowledges funding by a postdoc fellowship of the German Academic Exchange Service (DAAD).

1 Introduction

An *n-Venn diagram* is a collection of $n \geq 1$ simple closed curves in the plane that intersect in only finitely many points and that create exactly 2^n connected regions, one for every possible combination of being inside or outside with respect to each curve. It is easy to see that in any intersection point of at least two curves, at least two of them must cross, and thus we refer to any such intersection point as a *crossing*. A Venn diagram is *simple* if every crossing involves only two curves; see Figure 1.

© Sofia Brenner, Petr Gregor, Torsten Mütze, and Francesco Verciani;
licensed under Creative Commons License CC-BY 4.0



(a) $n = 3$ (b) $n = 4$ (c) $n = 5$

Figure 1 Simple Venn diagrams for $n = 3, 4, 5$, maximizing the number of crossings. For $n = 3, 4, 5$ the number of crossings $2^n - 2$ equals $6, 14, 30$. The diagrams (a)+(c) are rotationally symmetric. All diagrams are monotone and convex.

Venn diagrams are an appealing tool to visualize sets and their containment relations, and they are named after the English mathematician John Venn (1834–1923), who used them in the context of propositional logic [24]. Despite the simple definition, many questions about Venn diagrams lead to interesting and challenging mathematical and computational problems, which triggered a long and fruitful line of research devoted to them; see Ruskey and Weston's survey [19] and the many beautiful illustrations therein. These problems touch and connect various areas such as discrete geometry, graph drawing, graph theory, poset theory, coding theory, and enumerative combinatorics.

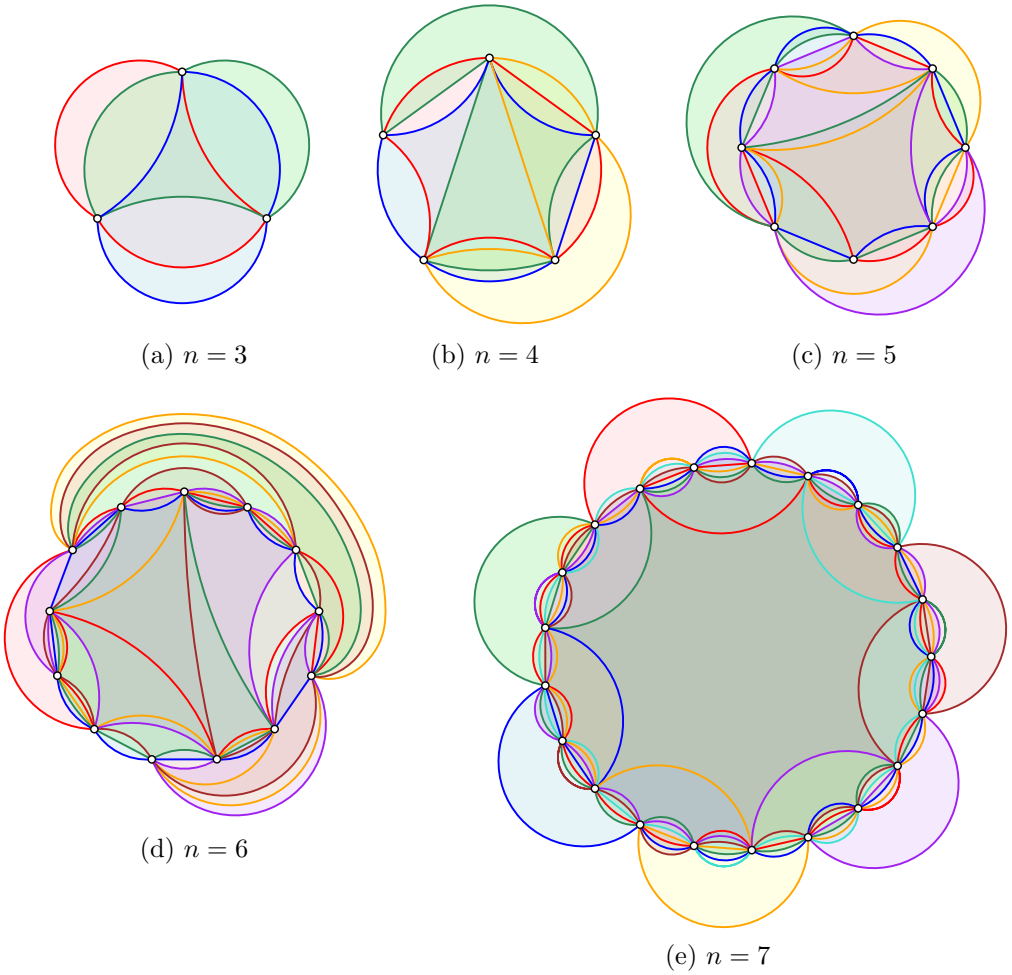
General constructions of n-Venn diagrams, valid for every $n \geq 1$, were provided already by Venn [24] and much later by Edwards [7, 8]. In fact, these two constructions are connected via the well-known binary reflected Gray code, a listing of all 2^n binary strings of length n such that any two consecutive strings differ in a single bit.

Particularly pleasing for the human eye are rotationally *symmetric* Venn diagrams, such as the ones in Figure 1 (a)+(c) and Figure 2 (a)+(e). They exist if and only if $n \geq 2$ is a prime number. The necessity of this condition was established by Henderson [13], and the sufficiency, i.e., a construction valid for all prime n was shown by Griggs, Killian and Savage [10]. Their approach builds a symmetric chain partition in the so-called necklace poset. The problem to find *simple* symmetric n-Venn diagrams is open in general. Solutions for small cases are only known for $n = 1, 3, 5, 7, 11, 13$ [16], and for general prime n a construction is known that guarantees at least half of all crossings to be simple [15].

A *k-region* in an n-Venn diagram is a region that lies inside of exactly k of the curves, and outside the remaining $n - k$ curves. A *monotone* diagram is one in which for every $0 < k < n$, every k-region is adjacent to both a $(k-1)$-region and a $(k+1)$-region. A *convex* diagram is one in which every curve is convex. It is not hard to show that convex diagrams are monotone. Bultena, Grünbaum and Ruskey [3] also proved the converse, namely that every monotone diagram is isomorphic to a convex one. Thus, the combinatorial notion of monotonicity completely captures the geometric notion of convexity.

The number of non-isomorphic simple n-Venn diagrams for $n = 1, \ldots, 6$ is $1, 1, 1, 1, 20$, $3\,430\,404$, and the number of monotone simple diagrams is $1, 1, 1, 1, 11, 32\,255$ [2, 6, 12] (OEIS A386795 and A390247, respectively).

A well-known conjecture in the area, raised by Peter Winkler [26] in 1984 and reiterated in [27, 28], was that every simple n-Venn diagram can be extended to a simple $(n+1)$-Venn diagram by adding a suitable curve. Very recently, Winkler's conjecture was disproved by Brenner, Kleist, Mütze, Rieck, and Verciani [2], who constructed counterexamples to the conjecture for all $n \geq 6$. In particular, out of the $3\,430\,404$ many 6-Venn diagrams, 72 are



(a) $n = 3$ (b) $n = 4$ (c) $n = 5$

(d) $n = 6$ (e) $n = 7$

Figure 2 Non-simple Venn diagrams for $n = 3, 4, 5, 6, 7$ that minimize the number of crossings. For those values of n, the number of crossings $L_n = \lceil \frac{2^n-2}{n-1} \rceil$ equals $3, 5, 8, 13, 21$. Only in the diagrams (a) and (e) every crossing involves all n curves. The diagrams (a) and (e) are symmetric. The diagram (a) is monotone, while (b)–(e) are not monotone. None of the diagrams is convex, though (a) can be made convex by fixing the crossings and moving the centers of the circular arcs.

not extendable to a 7-Venn diagram. Already earlier, Grünbaum [11] proposed a variant of Winkler's conjecture, by dropping the requirement for the diagrams to be simple. Grünbaum's conjecture was settled affirmatively by Chilakamarri, Hamburger and Pippert [5], using a classical theorem in graph theory of Whitney [25], later generalized by Tutte [23].

1.1 Maximizing and minimizing the number of crossings

In this work, we are particularly interested in the number of crossings in an n-Venn diagram. It is easy to see that simple n-Venn diagrams have exactly $2^n - 2$ many crossings, and this is the largest possible number of crossings among all n-Venn diagrams. To complement this, Bultena and Ruskey [4] considered Venn diagrams with the smallest possible number of crossings. They proved that any n-Venn diagram with $n \geq 2$ has at least

$$L_n := \left\lceil \frac{2^n - 2}{n - 1} \right\rceil$$

many crossings, and they called diagrams achieving this lower bound *minimum* Venn diagrams. If $(2^n - 2)/(n-1)$ is integral, which happens for $n = 2, 3, 7, 19, 43, 55, 127, 163, \ldots$ (this is OEIS sequence A014741 incremented by 1), then this means that every crossing of the diagram involves all n curves. So far, minimum Venn diagrams are only known for $n \leq 7$; see Figure 2. Bultena and Ruskey [4] conjectured that minimum n-Venn diagrams exist for all $n \geq 8$, and this is also mentioned as an open problem in Ruskey and Weston's survey [19].

As a partial result, Bultena and Ruskey [4] proved that among *monotone* n-Venn diagrams with $n \geq 2$, the smallest possible number of crossings is $\binom{n}{\lfloor n/2 \rfloor}$, and this lower bound is achievable. Note that $\binom{n}{\lfloor n/2 \rfloor} = \frac{2^n}{\sqrt{\pi n/2}}(1 + o(1))$, so this is still a $\Theta(\sqrt{n})$-factor away from the general lower bound L_n.

The problem of minimizing crossings in Venn diagrams can be seen as a variant of the well-known and heavily studied *crossing number problem* [9, 20], where one attempts to draw a graph in the plane with the goal of minimizing the number of edge crossings. Even for the complete graph K_n or the hypercube Q_n, this quantity is not known exactly.

1.2 Our results

Our first construction yields n-Venn diagrams for the case when n is a power of 2 in which the number of crossings is minimized up to a $(1 + o(1))$-factor (as $n \to \infty$).

▶ **Theorem 1.** *There is an* 8*-Venn diagram with* 40 *crossings, and for every* $k \geq 4$ *and* $n := 2^k$*, there is an* n*-Venn diagram with exactly*

$$\left(1 + \frac{33}{8n} - \frac{2}{2^{n/2}} - \frac{2n}{2^n}\right)\frac{2^n}{n} \leq \left(1 + \frac{33}{8n}\right) L_n = (1 + o(1))L_n$$

many crossings.

The 8-Venn diagram with 40 crossings is shown in Figure 3.

For a general number of curves, i.e., when n is not necessarily a power of 2, we apply a doubling construction to obtain an n-Venn diagram in which the number of crossings is minimized up to a $(2 + o(1))$-factor.

▶ **Theorem 2.** *For every* $0 \leq m \leq 7$ *there is an* $(8+m)$*-Venn diagram with* $40 \cdot 2^m$ *crossings, and for every* $k \geq 4$*,* $n := 2^k$ *and* $0 \leq m < n$*, there is an* $(n+m)$*-Venn diagram with exactly*

$$\left(1 + \frac{33}{8n} - \frac{2}{2^{n/2}} - \frac{2n}{2^n}\right)\frac{2^{n+m}}{n} \leq \left(1 + \frac{33}{8n}\right)\frac{n+m}{n} L_{n+m} \leq (2 + o(1))L_{n+m}$$

many crossings.

Figure 4 shows the approximation ratio guaranteed by Theorem 2 for all $8 \leq n + m \leq 255$.

In particular, Theorem 2 gives n-Venn diagrams with the smallest known number of crossings for all $n \geq 8$; see Table 1.

1.3 Duals of Venn diagrams

Venn diagrams are conveniently studied by considering their dual graph; see Figure 5. Specifically, we consider the *n-dimensional hypercube Q_n*, or *n-cube* for short, the graph formed by all subsets of $[n] := \{1, 2, \ldots, n\}$, with an edge between any two sets that differ in a single element. For an edge $e = \{x, x \cup \{i\}\}$ of Q_n, we refer to i as its *direction*.

Table 1 The number of crossings in the Venn diagrams obtained from Theorem 2 versus the lower bound, and the minimum numbers for monotone diagrams.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
lower bound $L_n = \lceil \frac{2^n-2}{n-1} \rceil$ [4]	0	2	3	5	8	13	21	37	64	114	205	373	683	1261	2341	4369
Figure 2	0	2	3	5	8	13	21									
Theorem 2								40	80	160	320	640	1280	2560	5120	5118
monotone $\binom{n}{\lfloor n/2 \rfloor}$ [4]	0	2	3	6	10	20	35	70	126	252	462	924	1716	3432	6435	12870

The dual graph $Q(D)$ of an n-Venn diagram D satisfies the following properties:

(1) It is a spanning subgraph of Q_n, i.e., all 2^n vertices are present. Specifically, the vertex in $Q(D)$ corresponding to a region of the diagram D is the set of all indices of curves that contain this region in their interior.

(2) It is a plane graph, i.e., it is drawn in the plane without edge crossings, such that each face has even length 2ℓ for some integer $2 \leq \ell \leq n$ and contains exactly two edges of ℓ distinct directions. These ℓ directions correspond to the ℓ curves intersecting in the crossing corresponding to the face.

(3) For every $j \in [n]$, the two subgraphs of $Q(D)$ induced by all vertices x with $j \in x$, and with $j \notin x$, respectively, are connected. This corresponds to the jth curve being simple.

Conversely, the dual of any subgraph of Q_n satisfying these properties is an n-Venn diagram.

Clearly, the number of crossings of the diagram D equals the number of faces of the dual graph $Q(D)$. If D is a minimum n-Venn diagram, then almost all crossings involve all n curves, i.e., in the dual graph $Q(D)$, almost all faces have length $2n$.

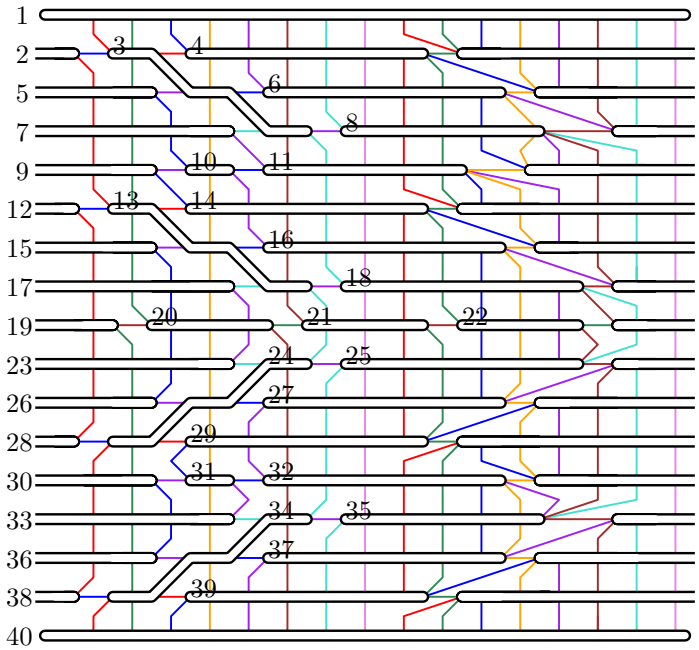


Figure 3 An 8-Venn diagram with 40 crossings obtained from Theorem 1. The 8 curves are drawn with colors. Each black/white bubble represents one crossing that has to be contracted to a single point, where the open bubbles at the left and right boundary wrap around at the bottom.

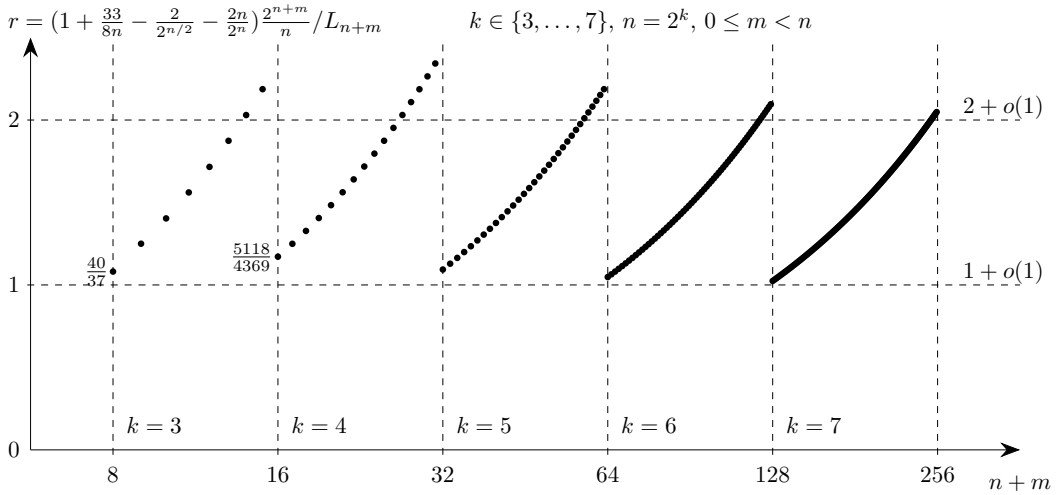


Figure 4 Log-linear plot of the ratio r of the number of crossings of the Venn diagrams obtained from Theorem 2 and the lower bound.

1.4 Proof ideas

Our proofs of Theorems 1 and 2 are constructive, and we proceed to give an informal sketch of the main steps of these constructions.

Using the characterization presented in the previous section, we build n-Venn diagrams by constructing planar spanning subgraphs of the hypercube Q_n satisfying conditions ①–③, with the goal of minimizing the number of faces.

For $n = 2^k$ we start with a partition of Q_n into isometric cycles of length $2n$ that is derived from a partition of Q_{n-1} into isometric paths found by Ramras [18]. These cycles are translates of a single isometric cycle of length $2n$ by a certain linear subspace of $\mathbb{Z}_2^n$ of dimension $d := 2^k - k - 1$. For our construction we choose a slightly different basis of this space, denoted by C_k, than originally used by Ramras.

In the first step of the construction, we embed the isometric cycles of the partition into the plane concentrically in an order that is specified by a certain Hamiltonian path R of Q_d (see Figure 9). Each vertex of Q_d represents coefficients of a linear combination of the basis C_k, so consecutive concentric cycles in this order differ only by one element of the basis C_k. This guarantees that there are edges between consecutive cycles that create mainly large faces, but also some 6-faces. These edges between pairs of consecutive cycles are added in the second step.

In the last step, some cycle edges shared by 6-faces are removed to further reduce the total number of faces. For this purpose we specify a Hamiltonian path R that contains long runs in its flip sequence. A *run* is a contiguous increasing or decreasing subsequence with increment or decrement 1, respectively. Interestingly, our construction of such long-run Hamiltonian cycles uses the same partition into isometric cycles as the aforementioned Venn diagram construction.

For values of n that are not powers of 2, we use a straightforward doubling construction (see Lemma 10) to derive n-Venn diagrams with relatively few crossings from the ones of the next smaller power of 2 constructed as described before. This is why the number of crossings increasingly deviates from the lower bound L_n the farther the distance from the next smaller power of 2 gets, until the next larger power of 2 resets the process, which explains the behavior seen in Figure 4.

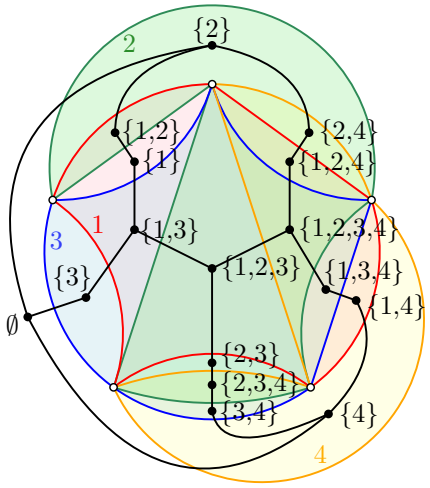


Figure 5 The dual of the 4-Venn diagram from Figure 2 (b) is a spanning subgraph of the 4-cube.

2 Isometric partitions of hypercubes

2.1 Preliminaries

Recall that $[n] = \{1, 2, \ldots, n\}$. We also define $2^{[n]} := \{x \mid x \subseteq [n]\}$. For $x \subseteq [n]$ we write $\overline{x} := [n] \setminus x$ for the complement of x w.r.t. the ground set $[n]$. Given a vertex x of Q_n, we refer to $\overline{x}$ as the *antipodal* vertex of x. For $x \subseteq [n]$ and an integer $k \geq 1$ we define $k \cdot x := \{ki \mid i \in x\}$. All these operations thread in the natural way over sets and sequences. For example, for a set $X \subseteq 2^{[n]}$ and an integer $k \geq 1$ we have $k \cdot X = \{k \cdot x \mid x \in X\}$.

For $x, y \subseteq [n]$ we define the symmetric difference as $x \oplus y := (x \setminus y) \cup (y \setminus x)$. For any set $X \subseteq 2^{[n]}$ we define the *span* of X by $\langle X \rangle := \{x_1 \oplus x_2 \oplus \cdots \oplus x_t \mid t \in \{0, \ldots, n\}$ and $x_i \in X$ for all $i \in [t]\}$.

We refer to an edge $e = \{x, x \oplus \{i\}\}$ of direction i in Q_n as an *i-edge*. The *flip sequence* of a path or cycle P in Q_n, denoted by $\sigma(P)$, is the sequence of directions of edges along P. We write $|\sigma(P)|$ for the length of this sequence. For example, the flip sequence of the path $P = (\{1,2\}, \{1,2,3\}, \{1,3\}, \{3\}, \{2,3\})$ in Q_3 is $\sigma(P) = (3, 2, 1, 2)$ and we have $|\sigma(P)| = 4$.

A subgraph H of a graph G is *isometric* if it preserves distances, i.e., $d_H(u, v) = d_G(u, v)$ for any two vertices $u, v \in V(H)$. In particular, a path in the hypercube Q_n is isometric if and only if it contains no two edges of the same direction. A cycle in Q_n is isometric if and only if the edges of the same direction come in pairs that lie oppositely on the cycle.

2.2 Partition into isometric paths

Ramras [18] described a partition of the hypercube Q_{n-1}, for $n = 2^k$ and $k \geq 1$, into isometric paths. For this we consider the sequence of sets $(B_k)_{k \geq 1}$ defined recursively by $B_1 := \emptyset$ and

$$B_k := B_{k-1} \cup \{\{1, 2^{k-1} + 1\}, \{2, 2^{k-1} + 2\}, \ldots, \{2^{k-1} - 1, 2^k - 1\}\} \text{ for } k \geq 2. \tag{1}$$

According to this definition, the first few sets are

$$B_1 = \emptyset, \qquad B_2 = \{\{1,3\}\}, \qquad B_3 = \{\{1,3\}, \{1,5\}, \{2,6\}, \{3,7\}\},$$
$$B_4 = \{\{1,3\}, \{1,5\}, \{2,6\}, \{3,7\}, \{1,9\}, \{2,10\}, \{3,11\}, \{4,12\}, \{5,13\}, \{6,14\}, \{7,15\}\}.$$

The sets in B_k, viewed as binary (characteristic) vectors, are linearly independent and hence they form the basis of a linear subspace $\langle B_k \rangle$ of dimension $|B_k| = 2^k - k - 1$ of the space $\mathbb{Z}_2^n$. For any $x \in \langle B_k \rangle$, Ramras [18] defines a path $P(x)$ of length $n-1$ in Q_{n-1} by

$$P(x) := (x, x \oplus \{1\}, x \oplus \{1,2\}, \ldots, x \oplus \{1,2,\ldots,n-1\}). \tag{2}$$

The flip sequence along this path is $\sigma(P) = (1, 2, \ldots, n-1)$, and the end vertices of $P(x)$ are antipodal in Q_{n-1}, i.e., the last vertex $x \oplus \{1,2,\ldots,n-1\}$ is the complement of the first vertex x w.r.t. the ground set $[n-1]$. The following theorem is illustrated on the left hand side of Figure 6.

▶ **Theorem 3** ([18, Thm. 2.5]). *For any $k \geq 1$ and $n := 2^k$, the isometric paths $P(x)$ of length $n-1$ defined in (2) for all $x \in \langle B_k \rangle$ form a partition of the vertex set of Q_{n-1}.*

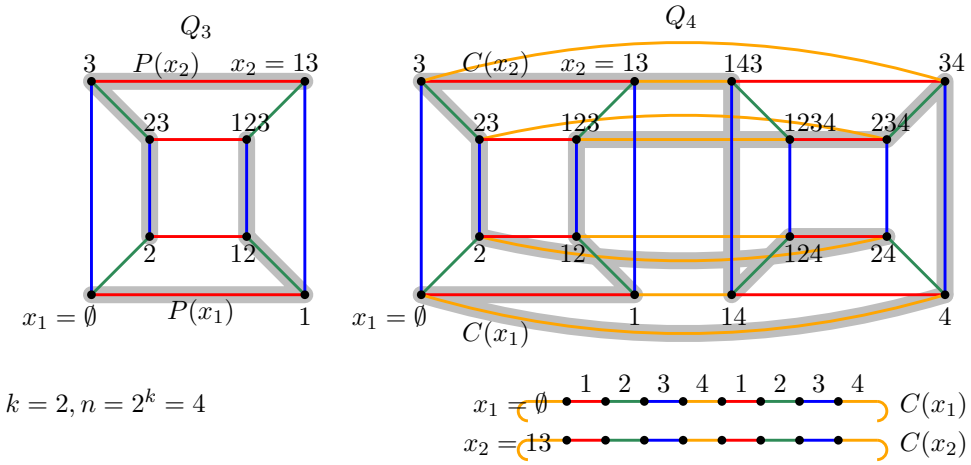


Figure 6 Illustration of Theorem 3 (left) and Corollary 5 (right) for the case $k = 2$. The bottom right shows a schematic illustration of the cycles showing only the starting vertices on the left and the flip sequences along each cycle.

2.3 Finding another basis for $\langle B_k \rangle$

For our purposes, we choose a different basis of the subspace $\langle B_k \rangle$, namely one that contains a larger number of 2-sets $\{a, b\}$ with $|b - a| = 2$. We define

$$\begin{aligned} &C_1 := O_1 := \emptyset, \\ &O_k := \{\{1,3\}, \{3,5\}, \ldots, \{2^k-3, 2^k-1\}\} \text{ for } k \geq 2, \text{ and} \\ &C_k := O_k \cup 2C_{k-1} \text{ for } k \geq 2. \end{aligned} \tag{3}$$

According to this definition, the first few sets are

$$\begin{aligned} &O_1 = \emptyset, \quad O_2 = \{\{1,3\}\}, \quad O_3 = \{\{1,3\}, \{3,5\}, \{5,7\}\}, \\ &O_4 = \{\{1,3\}, \{3,5\}, \{5,7\}, \{7,9\}, \{9,11\}, \{11,13\}, \{13,15\}\}, \end{aligned}$$

and

$$\begin{aligned} &C_1 = \emptyset, \quad C_2 = \{\{1,3\}\}, \quad C_3 = \{\{1,3\}, \{3,5\}, \{5,7\}, \{2,6\}\}, \\ &C_4 = \{\{1,3\}, \{3,5\}, \{5,7\}, \{7,9\}, \{9,11\}, \{11,13\}, \{13,15\}, \{2,6\}, \{6,10\}, \{10,14\}, \{4,12\}\}. \end{aligned}$$

Note that $|b-a| = 2$ for every $\{a,b\} \in O_k$ and furthermore, $|O_k| = 2^{k-1}-1$ and $|C_k| = 2^k - k - 1$.

▶ **Lemma 4.** *The sets B_k and C_k defined in (1) and (3), respectively, satisfy $\langle B_k \rangle = \langle C_k \rangle$ for any $k \geq 1$.*

The proof of Lemma 4 can be found in the preprint [1].

2.4 Partition into isometric cycles

From Theorem 3 it follows that for $n = 2^k$

$$C(x) := (P(x), \overline{P(x)} \cup \{n\}) \tag{4}$$

is an isometric cycle of length $2n$ in Q_n, with flip sequence $\sigma(C(x)) = (1,2,\ldots,n,1,2,\ldots,n)$, and furthermore, the cycles $C(x)$ for all $x \in \langle B_k \rangle$ form a partition of the vertex set of Q_n. Combining this with Lemma 4 we obtain the following isometric partition into cycles, which is the starting point of our constructions later on; see the right hand side of Figure 6.

▶ **Corollary 5.** *For any $k \geq 1$ and $n := 2^k$, the isometric cycles $C(x)$ of length $2n$ defined in (4) for all $x \in \langle C_k \rangle$ form a partition of the vertex set of Q_n.*

The following lemma describes the edges between the cycles $C(x)$, and it is illustrated in Figure 7. The proof is elementary, and we omit it.

▶ **Lemma 6.** *Let $k \geq 2$ and $n := 2^k$, and let $x, y \in \langle C_k \rangle$ be such that $y = x \oplus \{a,b\}$ with $a < b$. Then the cycles $C(x)$ and $C(y)$ have the following edges between them in Q_n:*

- **(i)** *From the start/end vertex of each of the two a-edges of $C(x)$, there is a b-edge to the end/start vertex of the corresponding a-edge of $C(y)$.*
- **(ii)** *From the start/end vertex of each of the two b-edges of $C(x)$, there is an a-edge to the end/start vertex of the corresponding b-edge of $C(y)$.*

If $b = a+2$, then in addition the following edges are present:

- **(iii)** *From the start/end vertex of each of the two a-/$(a+2)$-edges of $C(x)$, there is an $(a+1)$-edge to the end/start vertex of the corresponding $(a+2)$-/a-edge of $C(y)$.*

For $b > a+2$ we write $E(x,\{a,b\})$ for the set of four edges shown in Figure 7 (a1). Furthermore, for $b = a+2$ we write $E_{\diagdown}(x,a)$ and $E_{\diagup}(x,a)$ for the two sets of three edges shown in parts (b1) and (b2) of the figure, respectively. Lastly, we write $F(x,\{a,b\})$ for the 4-cycle shown in part (a2) of the figure.

3 Long-run Gray code construction

Given a path or cycle P in Q_n and integer $\rho \in [n]$, an *increasing* or *decreasing ρ-run* in its flip sequence $\sigma(P)$ is a contiguous subsequence $(a, a+1, a+2, \ldots, a+\ell)$ or $(a+\ell, a+\ell-1, \ldots, a+1, a)$ such that $a+\ell \leq \rho$. In words, it is a sequence of values that are increasing or decreasing (with increment 1 or decrement 1, respectively) such that all flipped directions are at most ρ. The parameter $\ell \geq 0$, which is one less than the length of the subsequence, is called the *length* of the run.

A ρ-run is *maximal* if it cannot be extended, i.e., if it is not contained in another ρ-run. Note that two maximal ρ-runs may overlap in at most one element, namely the last element of an increasing run and the first element of a decreasing run, or vice versa. Furthermore, note that a maximal ρ-run of length 0 does not overlap with any other ρ-runs, and a maximal ρ-run

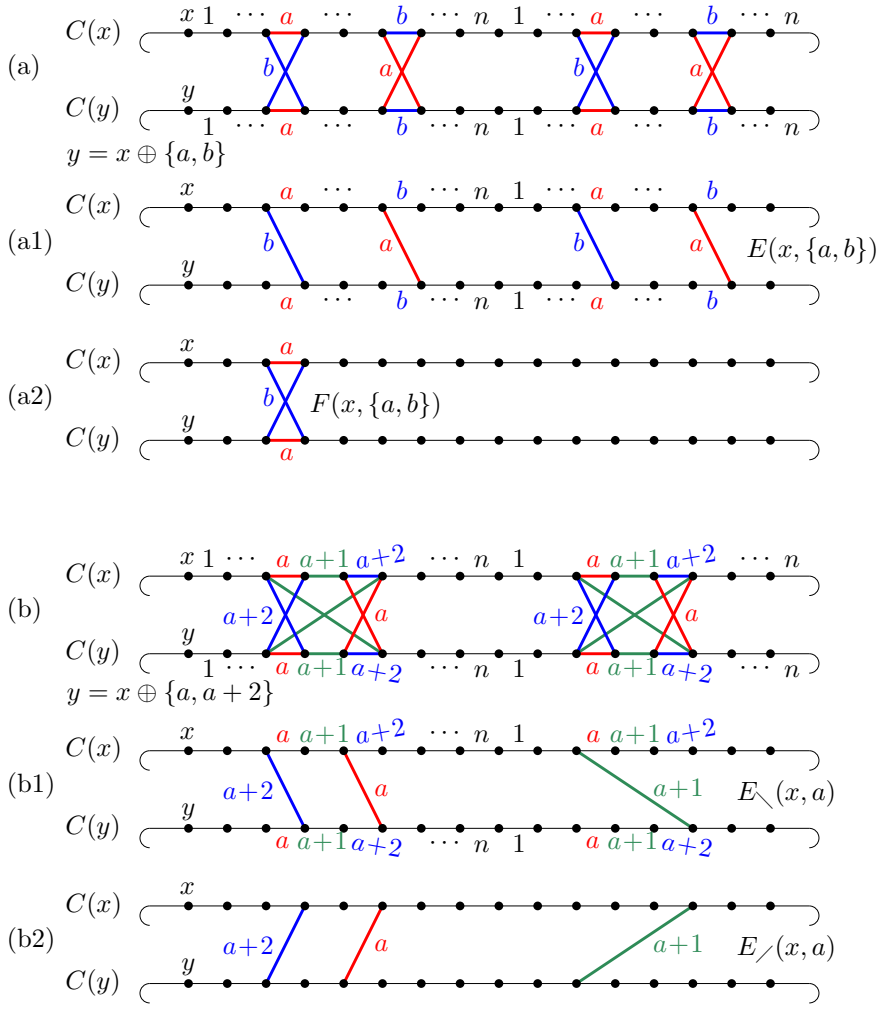


Figure 7 Illustration of Lemma 6 and definitions of the edge sets $E(x, \{a, b\})$, $F(x, \{a, b\})$, $E_{\diagdown}(x, a)$, and $E_{\diagup}(x, a)$.

of length 1 does not overlap with two other ρ-runs in the first and last element, as this would create a flip sequence $a+1, a, a+1, a$ or $a, a+1, a, a+1$, i.e., a 4-cycle, which is impossible. A *ρ-run partition* of $\sigma(P)$ is obtained by considering all maximal ρ-runs in $\sigma(P)$, and removing, from every pair of consecutive overlapping runs, the element in which they overlap from one of the two runs. For example, the flip sequence $\sigma(P) = (3, 1, 2, 3, 2, 1, 2, 4, 3, 2, 4)$ has the maximal 3-runs $(\overline{3}, 1, 2, \overline{3}, 2, \overline{1, 2}, \not{4}, \underline{3, 2}, \not{4})$, where increasing and decreasing runs are marked by overbrackets and underbrackets, respectively, and the cancelled elements are not contained in any 3-run. Thus we obtain $(\boxed{3}, \boxed{1, 2}, \boxed{3, 2, 1}, \boxed{2}, \not{4}, \boxed{3, 2}, \not{4})$ as a 3-run partition of $\sigma(P)$, where the boxes indicate the runs. Similarly, $(\boxed{3}, \boxed{1, 2, 3}, \boxed{2, 1}, \boxed{2}, \not{4}, \boxed{3, 2}, \not{4})$, $(\boxed{3}, \boxed{1, 2}, \boxed{3, 2}, \boxed{1, 2}, \not{4}, \boxed{3, 2}, \not{4})$ and $(\boxed{3}, \boxed{1, 2, 3}, \boxed{2}, \boxed{1, 2}, \not{4}, \boxed{3, 2}, \not{4})$ are also valid 3-run partitions.

Although there may be different ρ-run partitions, their number and total length, i.e., sum of lengths of all runs, is the same, and we denote these quantities by $\nu_\rho(P)$ and $\lambda_\rho(P)$. Note that the number of entries of $\sigma(P)$ exceeding ρ equals $|\sigma(P)| - \nu_\rho(P) - \lambda_\rho(P)$. In the example from before we have $\nu_3(P) = 5$ and $\lambda_3(P) = 4$.

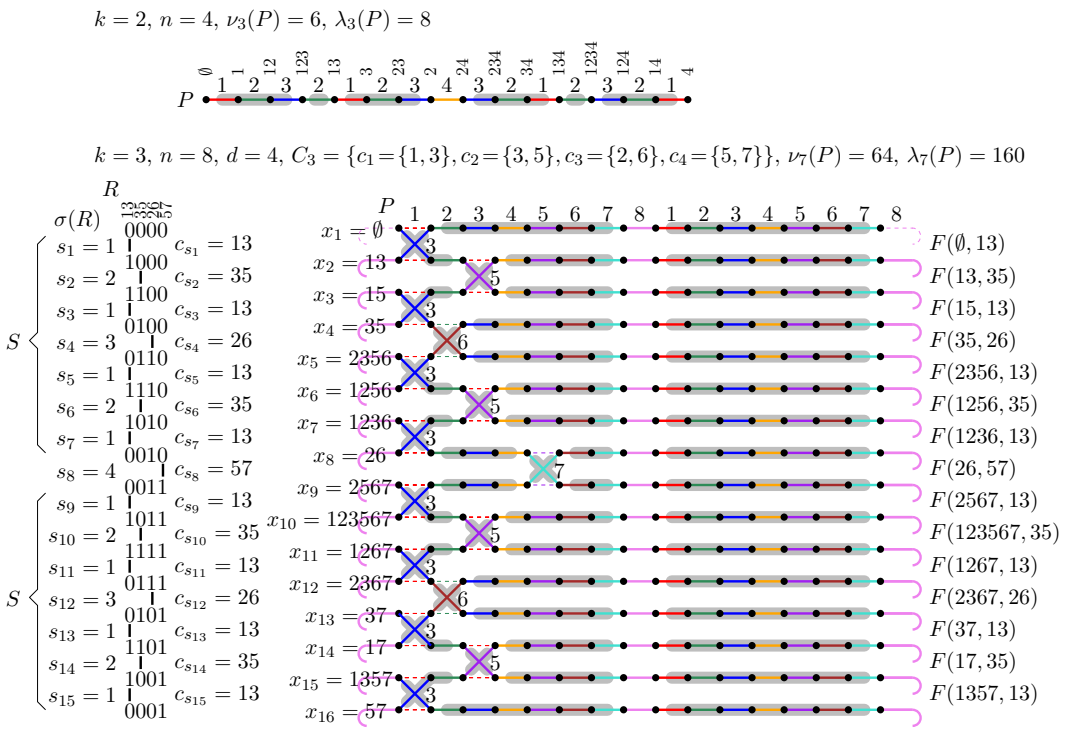


Figure 8 Illustration of the proof of Lemma 7. The edges that are removed while taking the symmetric difference are dashed, including the wrap-around edge of the first cycle. The subsequences highlighted in gray are $(n-1)$-runs.

Our first aim is to find a Hamiltonian path P in the hypercube Q_n for $n = 2^k$ for which the quantity $\nu_{n-1}(P) + 2\lambda_{n-1}(P)$ is as large as possible, i.e., we want to maximize the number of flipped directions contained in long $(n-1)$-runs. We shall see that maximizing this quantity corresponds to minimizing the number of crossings in the Venn diagrams constructed in the next section (see Lemma 9).

▶ **Lemma 7.** *Let $k \geq 2$ and $n := 2^k$. There is a Hamiltonian path P in Q_n that satisfies*

$$\nu_{n-1}(P) = \begin{cases} 6 & \text{if } k = 2, \\ \frac{17}{8} \cdot \frac{2^n}{n} - 4 & \text{if } k \geq 3, \end{cases} \quad \text{and } \lambda_{n-1}(P) = \begin{cases} 8 & \text{if } k = 2, \\ 2^n - \frac{25}{8} \cdot \frac{2^n}{n} + 4 & \text{if } k \geq 3. \end{cases}$$

Proof. For $k = 2$ we take the Hamiltonian path P with $\sigma(P) = (\boxed{1,2,3}, \boxed{2}, \boxed{1,2,3}, \boxed{4}, \boxed{3,2,1}, \boxed{2}, \boxed{3,2,1})$, proving that $\nu_3(P) = 6$ and $\lambda_3(P) = 8$; see the top part of Figure 8.

For $k \geq 3$ we construct a Hamiltonian path P in Q_n as follows; see the bottom part of Figure 8. We start with the partition of Q_n into cycles given by Corollary 5. Formally, we define the cycle factor $\mathcal{C} := \bigcup_{x \in \langle C_k \rangle} C(x)$. Each of the cycles of $\mathcal{C}$ has length $2n$ and the total number of cycles is $2^n/(2n) = 2^{2^k}/2^{k+1} = 2^{2^k-k-1} = 2^d$ for $d := 2^k - k - 1$. Recall that $|C_k| = 2^k - k - 1 = d$.

Let $(c_1, \ldots, c_d)$ be a sequence obtained by sorting all elements of the set C_k such that $c_1 = \{1,3\}, c_2 = \{3,5\}, c_3 = \{2,6\}$, and the remaining elements appear in arbitrary order. We take a Hamiltonian path R in Q_d whose flip sequence $\sigma(R) =: (s_1, \ldots, s_{2^d-1})$ alternates copies of the subsequence $S := (1,2,1,3,1,2,1)$ with single flips $s_{8i} \geq 4$ for $i \geq 1$, starting with S, i.e., $\sigma(R) = (S, s_8, S, s_{16}, S, s_{24}, \ldots, S, s_{2^d-8}, S)$. For example, we may take for R the

well-known binary reflected Gray code in Q_d. Note that $\sigma(R)$ contains exactly 2^{d-3} copies of S. We define an ordering $x_1, \dots, x_{2^d}$ of all elements of $\langle C_k \rangle$ recursively by $x_1 \coloneqq \emptyset$ and $x_{i+1} \coloneqq x_i \oplus c_{s_i}$ for $i = 1, \dots, 2^d - 1$. Note that this definition is valid since each element of $\langle C_k \rangle$ is a linear combination of the basis C_k, and R visits all d-tuples of $\{0,1\}$-coefficients.

Consider the symmetric difference of the edge sets of the cycle factor $\mathcal{C}$ with all 4-cycles $F(x_i, c_{s_i})$ for $i = 1, \dots, 2^d - 1$. Each of the 4-cycles has a pair of opposite edges in common with two consecutive cycles from the factor, and thus glues them together to a single cycle. Therefore, the resulting set of edges is a Hamiltonian cycle in Q_n. Removing one of its n-edges gives a Hamiltonian path P in Q_n. Note that the 4-cycles $F(x_i, c_{s_i})$ and $F(x_{i+1}, c_{s_{i+1}})$ are edge-disjoint for $i = 1, \dots, 2^d - 2$.

In the remainder of the proof we compute the quantities $\nu_{n-1}(P)$ and $\lambda_{n-1}(P)$. Recall from (2) and (4) that for each cycle $C(x_i)$, $i = 1, \dots, 2^d$, the flip sequence is $\sigma(C(x_i)) = (1, 2, \dots, n, 1, 2, \dots, n)$, i.e., it has two maximal $(n-1)$-runs of length $n-2$ each. Observe from Figure 8 that when gluing together the eight cycles $C(x_i)$ for which $s_i \in \{1,2,3\}$ belongs to a copy of S in $\sigma(R)$, the flip sequence of the resulting cycle has the $(n-1)$-run partition

$$\big(\boxed{3,2},\boxed{5},\boxed{4,\dots,n-1},\not{n},\boxed{1,\dots,n-1},\not{n},$$
$$\boxed{3},\boxed{6},\boxed{3,\dots,n-1},\not{n},\boxed{1,\dots,n-1},\not{n},$$
$$\boxed{3,2},\boxed{5},\boxed{4,\dots,n-1},\not{n},\boxed{1,\dots,n-1},\not{n},$$
$$\boxed{3},\boxed{2,\dots,n-1},\not{n},\boxed{1,\dots,n-1},\not{n},$$
$$\boxed{3,2},\boxed{5},\boxed{4,\dots,n-1},\not{n},\boxed{1,\dots,n-1},\not{n},$$
$$\boxed{3},\boxed{6},\boxed{3,\dots,n-1},\not{n},\boxed{1,\dots,n-1},\not{n},$$
$$\boxed{3,2},\boxed{5},\boxed{4,\dots,n-1},\not{n},\boxed{1,\dots,n-1},\not{n},$$
$$\boxed{3},\boxed{2,\dots,n-1},\not{n},\boxed{1,\dots,n-1},\not{n}\big),$$

consisting of 30 runs of total length $16n - 46$.

When taking the symmetric difference with the $2^{d-3} - 1$ remaining 4-cycles $F(x_i, c_{s_i})$ with $s_i \geq 4$, then inserting each such 4-cycle splits two runs $\boxed{2,\dots,n-1}$ into two times two runs $\boxed{2,\dots,a-1}$, $\boxed{a+1,\dots,n-1}$ and two trivial runs $\boxed{b}$, where $\{a,b\} \coloneqq c_{s_i}$ such that $a < b$, i.e., the number of runs increases by 4 and the total length decreases by 4.

Lastly, note that removing one n-edge from the resulting Hamiltonian cycle to break it into the Hamiltonian path P does not change any $(n-1)$-runs.

Combining these observations, we obtain

$$\nu_{n-1}(P) = 30 \cdot 2^{d-3} + 4 \cdot (2^{d-3} - 1) = \frac{17}{8} \cdot \frac{2^n}{n} - 4 \quad \text{and}$$
$$\lambda_{n-1}(P) = (16n - 46) \cdot 2^{d-3} - 4 \cdot (2^{d-3} - 1) = 2^n - \frac{25}{8} \cdot \frac{2^n}{n} + 4,$$

where we used the relation $2^d = \frac{2^n}{2n}$. ◀

For a general dimension that is not necessarily a power of 2 we generalize the result from before by applying a straightforward product construction.

▶ **Corollary 8.** *Let $k \geq 2$, $n \coloneqq 2^k$, and $0 \leq m < n$. There is a Hamiltonian path P in Q_{n+m} that satisfies*

$$\nu_{n-1}(P) = \begin{cases} 6 \cdot 2^m & \text{if } k = 2, \\ 2^m\big(\frac{17}{8} \cdot \frac{2^n}{n} - 4\big) & \text{if } k \geq 3, \end{cases} \text{ and } \lambda_{n-1}(P) = \begin{cases} 8 \cdot 2^m & \text{if } k = 2, \\ 2^m\big(2^n - \frac{25}{8} \cdot \frac{2^n}{n} + 4\big) & \text{if } k \geq 3. \end{cases}$$

Proof. We use the fact that Q_{n+m} is isomorphic to the Cartesian product $Q_n \,\square\, Q_m$. Consequently, if P is a Hamiltonian path in Q_n and R is a Hamiltonian path in Q_m with flip sequence $\sigma(R) =: (s_1, \dots, s_{2^m-1})$, then Q_{n+m} has a Hamiltonian path P' with flip sequence

$$\sigma(P') = \big(\sigma(P), s_1+n, \mathrm{rev}(\sigma(P)), s_2+n, \sigma(P), \dots, s_{2^m-2}+n, \sigma(P), s_{2^m-1}+n, \mathrm{rev}(\sigma(P))\big),$$

where $\mathrm{rev}(\sigma(P))$ denotes the reverse of the sequence $\sigma(P)$. By taking the Hamiltonian path P in Q_n given by Lemma 7 and any Hamiltonian path R in Q_m, the lemma follows. ◀

Note that as $n \to \infty$, we have $\lambda_{n-1}(P) = (1-o(1))2^{n+m}$, i.e., almost all flips along the Hamiltonian path P belong to an $(n-1)$-run.

4 Proof of Theorem 1

In this section, we describe our constructions of almost-minimum n-Venn diagrams for the case where n is a power of 2, thus proving Theorem 1.

▶ **Lemma 9.** *Let $k \geq 3$, $n := 2^k$, $d := 2^k - k - 1$, and $\rho := 2^{k-1} - 1 = n/2 - 1$, and let P be a Hamiltonian path in Q_d. Then there is an n-Venn diagram with exactly $2 \cdot \frac{2^n}{n} - \nu_\rho(P) - 2\lambda_\rho(P) - 2$ many crossings.*

Proof. To prove the lemma, we construct a plane subgraph $H \subseteq Q_n$ satisfying properties ①–③ stated in Section 1.3 that has exactly $2 \cdot \frac{2^n}{n} - \nu_\rho(P) - 2\lambda_\rho(P) - 2$ many faces, as follows. We start with the partition of Q_n into cycles given by Corollary 5. Formally, we define the cycle factor $\mathcal{C} := \bigcup_{x \in \langle C_k \rangle} C(x)$. Each of the cycles of $\mathcal{C}$ has length $2n$ and the total number of cycles is $2^n/(2n) = 2^{2^k}/2^{k+1} = 2^{2^k-k-1} = 2^d$. Recall that $|C_k| = 2^k - k - 1 = d$.

Let $c = (c_1, \dots, c_d)$ be a sequence obtained by sorting all elements of the set C_k such that all 2-sets from $O_k \subseteq C_k$ appear first and with increasing minimum values, and the remaining 2-sets appear afterwards in arbitrary order. That is, we choose the first ρ elements as $(c_1, \dots, c_\rho) = (\{1,3\}, \{3,5\}, \dots, \{2^k-3, 2^k-1\})$, which is the order from (3), or equivalently, $c_i = \{2i-1, 2i+1\}$ for $1 \leq i \leq \rho$. Recall that $|O_k| = 2^{k-1} - 1 = \rho \leq d$.

We consider the flip sequence $\sigma(P) =: (s_1, \dots, s_{2^d-1})$ of the given Hamiltonian path P in Q_d. The ρ-runs in this sequence will be particularly relevant for our construction. We define an ordering $x_1, \dots, x_{2^d}$ of all elements of $\langle C_k \rangle$ recursively by $x_1 := \emptyset$ and $x_{i+1} := x_i \oplus c_{s_i}$ for $i = 1, \dots, 2^d - 1$. Note that this definition is valid since each element of $\langle C_k \rangle$ is a linear combination of the basis C_k, and P visits all d-tuples of $\{0,1\}$-coefficients.

To construct H, we first embed the cycles of the factor $\mathcal{C}$ in the plane by nesting them concentrically according to this ordering, i.e., $C(x_1)$ is the outermost cycle, and for $i = 1, \dots, 2^d - 1$, the cycle $C(x_{i+1})$ is nested concentrically inside $C(x_i)$.

Next, we add edges between every pair of consecutive cycles $C(x_i)$ and $C(x_{i+1})$ for $i = 1, \dots, 2^d - 1$ according to the following rules; see Figure 9. We first fix a ρ-run partition of P throughout, so whenever we refer to a ρ-run in the following, we mean a run in this fixed partition. If $s_i \leq \rho$, then we have $c_{s_i} = \{a, a+2\}$ for $a := 2s_i - 1$. If the entry s_i of $\sigma(P)$ is contained in an increasing ρ-run, then we add the three edges of $E_\diagdown(x_i, a)$, and if it is contained in a decreasing ρ-run, then we add the three edges of $E_\diagup(x_i, a)$ instead. If the run has length 0, i.e., it consists only of a single entry, we treat it as one of the two cases, say, increasing. On the other hand, if $s_i > \rho$, then we add the four edges of $E(x_i, c_{s_i})$.

In this way, we obtain an intermediate connected plane graph with $C(x_1)$ as the outer face and $C(x_{2^d})$ as the innermost face that has either 3 or 4 faces between cycles $C(x_i)$ and $C(x_{i+1})$ for $i = 1, \dots, 2^d - 1$, namely if $s_i \leq \rho$ or $s_i > \rho$ holds, respectively. Since the

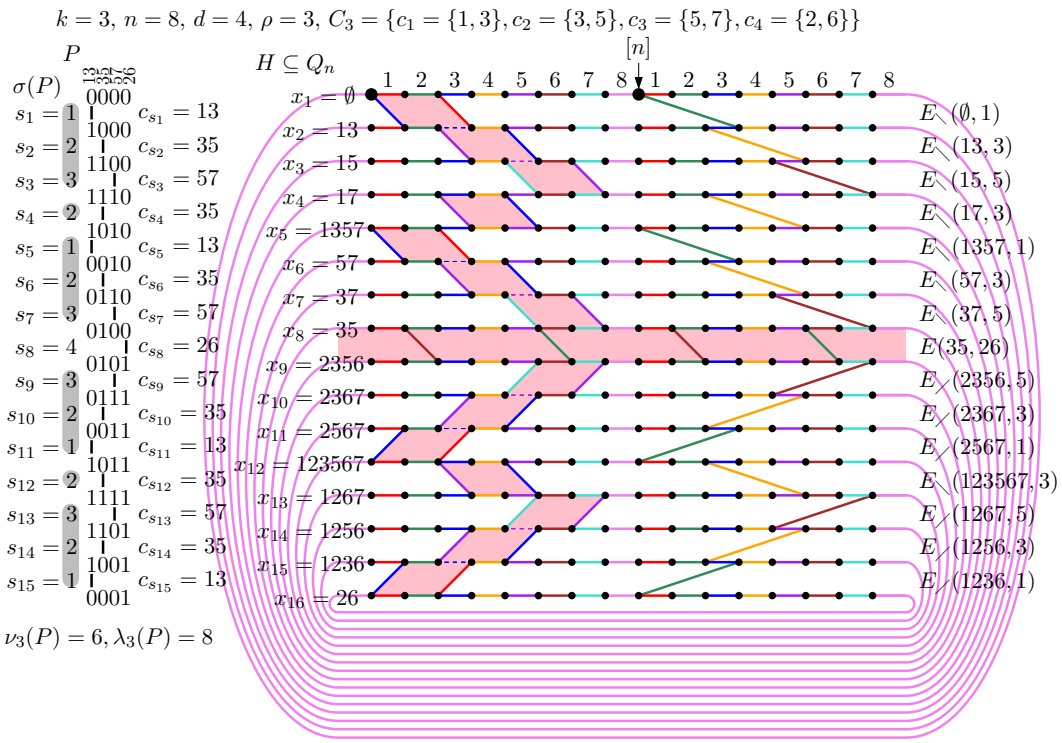


Figure 9 Illustration of the proof of Lemma 9 for the case $k = 3$, i.e., $n = 8$. The dashed edges are deleted, so they are not part of H. The constructed dual graph H of an 8-Venn diagram has 40 faces, i.e., the corresponding Venn diagram has 40 crossings; see Figure 3, only 3 more than the lower bound $L_8 = 37$. All faces except the red ones have maximum possible length $2n$.

number of entries s_i in $\sigma(P)$ with $s_i \le \rho$ is exactly $\nu_\rho(P) + \lambda_\rho(P)$, the number of faces of the intermediate graph is exactly

$$2 + 4(2^d - 1) - (\nu_\rho(P) + \lambda_\rho(P)) = 2 \cdot \frac{2^n}{n} - \nu_\rho(P) - \lambda_\rho(P) - 2.$$

In the last step of constructing H, we remove edges from some of the cycles $C(x_i)$, $i = 2, \dots, 2^d - 1$, according to the following rules: If s_{i-1} and s_i belong to the same increasing ρ-run in $\sigma(P)$, then we have $c_{s_{i-1}} = \{a-2, a\}$ and $c_{s_i} = \{a, a+2\}$ for $a := 2s_i - 1$, and then we remove the first a-edge of $C(x_i)$. If s_{i-1} and s_i belong to the same decreasing ρ-run in $\sigma(P)$, then we have $c_{s_{i-1}} = \{a, a+2\}$ and $c_{s_i} = \{a-2, a\}$ for $a := 2s_i + 1$, and then we remove the first a-edge of $C(x_i)$. This completes the description of how to construct the subgraph H of Q_n.

Note that each removed edge decreases the number of faces by 1. Since the number of removed edges is exactly $\lambda_\rho(P)$, the number of faces of H is exactly

$$2 \cdot \frac{2^n}{n} - \nu_\rho(P) - 2\lambda_\rho(P) - 2,$$

as claimed. It remains to check that the graph $H \subseteq Q_n$ just defined satisfies conditions ①–③.

Condition ① holds since the cycles $C(x_i)$ for $i = 1, \dots, 2^d$ form a partition of Q_n by Corollary 5. To verify condition ②, first observe in Figure 7 (a1), (b1) and (b2) that no two edges from any of the sets $E(x, \{a, b\})$, $E_\diagdown(x, a)$, or $E_\diagup(x, a)$ are crossing and thus

H is a plane graph. Furthermore, observe from Figure 7 (a1) that the four facial cycles between two cycles obtained by adding the edge set $E(x, \{a, b\})$ have lengths $2(b-a)+2$ and $2(n-(b-a))+2$ and flip sequences

$$(a, a+1, \ldots, b-1, a, b, b-1, \ldots, a+1, b), \text{ and}$$
$$(b, b+1, \ldots, n, 1, 2, \ldots, a-1, b, a, a-1, \ldots, 1, n, n-1, \ldots, b+1, a),$$

respectively. Similarly, observe from Figures 7 (b1) and (b2) that the three facial cycles between two cycles obtained by adding the edge set $E_{\diagdown}(x, a)$ or $E_{\diagup}(x, a)$ have lengths 6, $2n$, and $2n$ and flip sequences

$$(a, a+1, a, a+2, a+1, a+2),$$
$$(a+2, a+3, \ldots, n, 1, 2, \ldots, a-1, a+1, a+2, a+1, \ldots, 1, n, n-1, \ldots, a+3, a), \text{ and}$$
$$(a, a+1, \ldots, n, 1, 2, \ldots, a-1, a+2, a, a-1, \ldots, 1, n, n-1, \ldots, a+3, a+1),$$

respectively. It can be checked directly that the aforementioned faces, as well as the outer face $C(x_1)$ and the innermost face $C(x_{2^d})$ satisfy condition ②.

In the last step of the construction we removed some edges from the cycles $C(x_i)$. Specifically, for each increasing or decreasing run of length $\ell \geq 2$, we glue together ℓ adjacent 6-cycles, creating a new facial cycle of length $4\ell+2$. Observe that the adjacent 6-faces have edge directions shifted by 2 and therefore their directions overlap only in the largest resp. smallest direction, and the shared direction is the direction of the edge that was removed. It follows that the resulting cycles of length $4\ell+2$ also satisfy condition ②.

To prove condition ③ we establish by induction on $i = 1, \ldots, 2^d$ that for every direction $j \in [n]$ every vertex x of $C(x_i)$ is connected in H to either the vertex $\emptyset$ or $[n]$ along a path that avoids any j-edges. Specifically, x is connected to $\emptyset$ if $j \notin x$ and to $[n]$ if $j \in x$.

To settle the induction basis $i = 1$, recall that the flip sequence of $C(x_1)$ is $\sigma(C(x_1)) = (1, 2, \ldots, n, 1, 2, \ldots, n)$, so removing any two j-edges for some $j \in [n]$ from $C(x_1)$ results in two antipodal paths of length $n-1$, respectively containing vertex $\emptyset$ and vertex $[n]$.

For the induction step $i \to i+1$, it suffices to show that every vertex of $C(x_{i+1})$ is connected in H to some vertex of $C(x_i)$ along a path that has no j-edges. We distinguish two cases according to which edges were added between $C(x_i)$ and $C(x_{i+1})$.

In the case when the four edges of $E(x_i, c_{s_i})$, $c_{s_i} =: \{a, b\}$, were added between $C(x_i)$ and $C(x_{i+1})$, observe that the two added a-edges are incident to antipodal vertices of $C(x_{i+1})$, and the same is true for the two added b-edges. Hence removing any two j-edges, where $j \in [n]$, from $C(x_{i+1})$ results in two subpaths that are both connected to $C(x_i)$ along an edge of $E(x_i, c_{s_i})$ that is not a j-edge, even if $j \in \{a, b\}$.

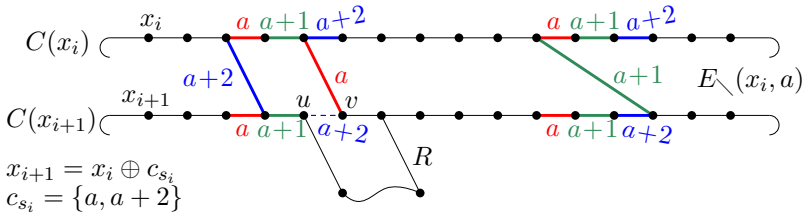


Figure 10 Verification of condition ③ in the proof of Lemma 9.

Now we consider the case when the three edges of $E_{\diagdown}(x_i, a)$ were added between $C(x_i)$ and $C(x_{i+1})$ (i.e., we have $b = a+2$). The third case when the edges of $E_{\diagup}(x_i, a)$ were added is analogous. We will also assume w.l.o.g. that the first $(a+2)$-edge on $C(x_{i+1})$ was

removed; otherwise the situation is even simpler; see Figure 10. We denote the removed $(a+2)$-edge by $\{u, v\}$. If $j \notin \{a, a+1, a+2\}$, then removing the j-edges from $C(x_{i+1})$ results in three subpaths that are all connected to $C(x_i)$, each path along an edge of $E_\diagdown(x_i, a)$. If $j = a+2$, then removing also the second $(a+2)$-edge from $C(x_{i+1})$ results in two subpaths of $C(x_{i+1})$ that are both connected to $C(x_i)$, since the added edges of directions a and $a+1$ are incident to antipodal vertices of $C(x_{i+1})$. For the last case $j \in \{a, a+1\}$ recall that there is a path R between u and v along the face of H containing these two vertices and vertices of $C(x_{i+j})$ for $j \geq 2$, and the path R does not contain edges of directions a or $a+1$. The path R and the two edges of $E_\diagdown(x_i, a)$ that are not of direction j connect all vertices of $C(x_{i+1})$ to $C(x_i)$ along a path not containing any j-edges. This shows that the graph H indeed satisfies condition ③. ◀

With Lemma 9 in hand, we are now in position to prove Theorem 1.

Proof of Theorem 1. For $k \geq 3$ we define $n := 2^k$, $d := 2^k - k - 1$ and $\rho := 2^{k-1} - 1 = n/2 - 1$.

If $k = 3$, we have $n = 8$, $d = 4$ and $\rho = 3$. We apply Lemma 7 for $\widehat{k} := k - 1 = 2$, $\widehat{n} := 2^{\widehat{k}} = 4$ to obtain a Hamiltonian path P in Q_4 with $\nu_3(P) = 6$ and $\lambda_3(P) = 8$. Therefore, Lemma 9 yields an 8-Venn diagram with exactly $2 \cdot \frac{2^8}{8} - 6 - 2 \cdot 8 - 2 = 40$ crossings; see Figures 3 and 9.

If $k \geq 4$, we apply Corollary 8 for $\widehat{k} := k - 1$, $\widehat{n} := 2^{\widehat{k}} = 2^{k-1} = n/2$ and $\widehat{m} := d - \widehat{n} = 2^{k-1} - k - 1$ to obtain a Hamiltonian path P in $Q_{\widehat{n}+\widehat{m}} = Q_d$ with

$$\nu_\rho(P) = \nu_{\widehat{n}-1}(P) = \left(\frac{17}{8} \cdot \frac{2^{\widehat{n}}}{\widehat{n}} - 4\right) 2^{\widehat{m}} = \frac{17}{8} \cdot \frac{2^n}{n^2} - \frac{2^{n/2}}{n/2} \quad \text{and}$$

$$\lambda_\rho(P) = \lambda_{\widehat{n}-1}(P) = \left(2^{\widehat{n}} - \frac{25}{8} \cdot \frac{2^{\widehat{n}}}{\widehat{n}} + 4\right) 2^{\widehat{m}} = \frac{2^n}{2n} - \frac{25}{8} \cdot \frac{2^n}{n^2} + \frac{2^{n/2}}{n/2}.$$

Therefore, Lemma 9 yields an n-Venn diagram with exactly

$$2 \cdot \frac{2^n}{n} - \nu_\rho(P) - 2\lambda_\rho(P) - 2 = \left(1 + \frac{33}{8n} - \frac{2}{2^{n/2}} - \frac{2n}{2^n}\right) \frac{2^n}{n}$$

many crossings.

Finally, note that the inequality $\frac{2^n}{n} \leq L_n = \lceil \frac{2^n - 2}{n-1} \rceil$ holds for all $n \geq 2$. ◀

5 Proof of Theorem 2

We now show how to lift the construction of n-Venn diagrams for the case when n is a power of 2 to the general case, thus proving Theorem 2. The lifting is achieved via the following straightforward doubling construction.

We say that a face of the dual graph of an n-Venn diagram is *colorful* if it has length $2n$ and contains two antipodal vertices of Q_n. In the primal Venn diagram, such a face corresponds to a crossing involving all n curves for which the cyclic ordering of curves around the crossing can be split into two halves such that every curve appears exactly once in each half. Consequently, there is a way to draw an additional curve through this crossing that crosses (not only intersects) each of the n existing curves.

▶ **Lemma 10.** *If there is an n-Venn diagram D whose dual graph has a colorful face, then there is an $(n+1)$-Venn diagram whose dual graph has a colorful face and with twice as many crossings as D.*

The proof of Lemma 10 can be found in the preprint [1]. With Lemma 10 in hand, we are now in position to prove Theorem 2.

Proof of Theorem 2. The duals of the Venn diagrams from Theorem 1 all have a colorful face, namely the outer face $C(x_1)$, so applying Lemma 10 (m times) proves the theorem. ◀

6 Remarks and open problems

We conclude with some remarks and challenging open problems.

- Is there a minimum 8-Venn diagram, i.e., one with only 37 crossings? The best one we found has 40 crossings; see Figure 3.
- Lemma 7 gives a Hamiltonian path P in Q_n, $n = 2^k$, with

$$\nu_{n-1}(P) + 2\lambda_{n-1}(P) = 2 \cdot 2^n - \frac{33}{8} \cdot \frac{2^n}{n} + 4.$$

  Is there another construction that would decrease the constant $\frac{33}{8}$? If so, then this would directly improve the same constant in Theorems 1 and 2. One can easily prove an upper bound of $\nu_{n-1}(P) + 2\lambda_{n-1}(P) \le 2 \cdot 2^n - \frac{2^n}{n}$.
- A long-standing problem due to Slater [21] is whether there exists a Hamiltonian path P in Q_n such that any two consecutive entries of the flip sequence differ by ± 1. Put differently, if we write $\mu(P)$ for the number of such consecutive entries of $\sigma(P)$, then the goal is to find a Hamiltonian path P with $\mu(P) = 2^n - 2$. While such a path exists for $n \le 6$, it does not exist for $n = 7$ [22] and $n = 8$ (computer experiments by Dimitrov, Gregor and Lužar; see [17, Problem 10]). We clearly have $\mu(P) \ge \lambda_n(P) \ge \lambda_{n-1}(P)$, so Lemma 7 and Corollary 8 yield a Hamiltonian path P for which $\mu(P) = (1-o(1)) \cdot (2^n - 2)$, which can be seen as an approximate solution to Slater's problem (cf. [14]).

References

1 Sofia Brenner, Petr Gregor, Torsten Mütze, and Francesco Verciani. On minimum Venn diagrams, 2025. arXiv:2511.09230.

2 Sofia Brenner, Linda Kleist, Torsten Mütze, Christian Rieck, and Francesco Verciani. Disproving two conjectures on the Hamiltonicity of Venn diagrams, 2025. arXiv:2503.18554.

3 Bette Bultena, Branko Grünbaum, and Frank Ruskey. Convex drawings of intersecting families of simple closed curves. In *Canadian Conference on Computational Geometry (CCCG)*, 1999. URL: http://www.cccg.ca/proceedings/1999/c14.pdf.

4 Bette Bultena and Frank Ruskey. Venn diagrams with few vertices. *Electron. J. Combin.*, 5:Research Paper 44, 21 pp., 1998. doi:10.37236/1382.

5 Kiran B. Chilakamarri, Peter Hamburger, and Raymond E. Pippert. Venn diagrams and planar graphs. *Geom. Dedicata*, 62(1):73–91, 1996. doi:10.1007/BF00240003.

6 Kiran B. Chilakamarri, Peter Hamburger, and Raymond E. Pippert. Analysis of Venn diagrams using cycles in graphs. *Geom. Dedicata*, 82(1-3):193–223, 2000. doi:10.1023/A:1005288325189.

7 Anthony W. F. Edwards. Venn diagrams for many sets. In *Bulletin of the International Statistical Institute, 47th Session, Paris, Book 1*, pages 311–312. IIS, 1989.

8 Anthony W. F. Edwards. Venn diagrams for many sets. *New Scientist*, 121(1646):51–56, 1989.

9 Paul Erdős and Richard K. Guy. Crossing number problems. *Amer. Math. Monthly*, 80:52–58, 1973. doi:10.2307/2319261.

10 Jerrold R. Griggs, Charles E. Killian, and Carla D. Savage. Venn diagrams and symmetric chain decompositions in the boolean lattice. *Electron. J. Comb.*, 11(1), 2004. doi:10.37236/1755.

11 Branko Grünbaum. Venn diagrams. I. *Geombinatorics*, 1(4):5–12, 1992.

12 Peter Hamburger and Raymond E. Pippert. Simple, reducible Venn diagrams on five curves and Hamiltonian cycles. *Geom. Dedicata*, 68(3):245–262, 1997. doi:10.1023/A:1004924423245.

13 David W. Henderson. Classroom Notes: Venn Diagrams for More than Four Classes. *Amer. Math. Monthly*, 70(4):424–426, 1963. doi:10.2307/2311865.

14 Gabriel S. Himelfarb and Moshe Schwartz. Improved constructions of skew-tolerant gray codes. *IEEE Trans. Inform. Theory*, 71(10):8017–8028, 2025. doi:10.1109/tit.2025.3592490.

15 Charles E. Killian, Frank Ruskey, Carla D. Savage, and Mark Weston. Half-simple symmetric Venn diagrams. *Electron. J. Comb.*, 11(1), 2004. doi:10.37236/1839.

16 Khalegh Mamakani and Frank Ruskey. New roses: Simple symmetric Venn diagrams with 11 and 13 curves. *Discret. Comput. Geom.*, 52(1):71–87, 2014. doi:10.1007/S00454-014-9605-6.

17 Torsten Mütze. Combinatorial Gray codes—an updated survey. *Electron. J. Combin.*, DS26(Dynamic Surveys):93 pp., 2023. doi:10.37236/11023.

18 Mark Ramras. Symmetric vertex partitions of hypercubes by isometric trees. *J. Combin. Theory Ser. B*, 54(2):239–248, 1992. doi:10.1016/0095-8956(92)90055-3.

19 Frank Ruskey and Mark Weston. A survey of Venn diagrams. *Electron. J. Combin.*, 4(1):Dynamic Survey 5, 1997. doi:10.37236/26.

20 Marcus Schaefer. The graph crossing number and its variants: a survey. *Electron. J. Combin.*, DS21(Dynamic Surveys):90 pp., 2013. doi:10.37236/2713.

21 Peter J. Slater. A problem concerning restricted Gray codes, open problem session. *Congr. Numer.*, XXIII–XIV:918–919, 1979. Proceedings of the Tenth Southeastern Conference on Combinatorics, Graph Theory and Computing (Boca Raton, FL, 1979).

22 Peter J. Slater. Research problems 109 and 110. *Discrete Math.*, 76(3):293–294, 1989. doi:10.1016/0012-365X(89)90330-0.

23 William T. Tutte. A theorem on planar graphs. *Trans. Amer. Math. Soc.*, 82:99–116, 1956. doi:10.2307/1992980.

24 John Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *Phil. Mag. S. 5.*, 9(59):1–18, 1880. doi:10.1080/14786448008626877.

25 Hassler Whitney. A theorem on graphs. *Ann. of Math. (2)*, 32(2):378–390, 1931. doi:10.2307/1968197.

26 Peter Winkler. Venn diagrams: Some observations and an open problem. *Congr. Numer.*, 45:267–274, 1984.

27 Peter Winkler. *Mathematical puzzles: A connoisseur's collection.* AK Peters Series, Taylor & Francis, 2004.

28 Peter Winkler. Puzzled: Where sets meet (Venn diagrams). *Commun. ACM*, 55(2):128, 2012. doi:10.1145/2076450.2076475.

Disproving Two Conjectures on the Hamiltonicity of Venn Diagrams

Sofia Brenner
Institute of Mathematics, University of Kassel, Germany

Linda Kleist
Department of Informatics, University of Hamburg, Germany

Torsten Mütze
Institute of Mathematics, University of Kassel, Germany

Christian Rieck
Institute of Mathematics, University of Kassel, Germany

Francesco Verciani
Institute of Mathematics, University of Kassel, Germany

Abstract

In 1984, Winkler conjectured that every simple Venn diagram with n curves can be extended to a simple Venn diagram with $n+1$ curves. This conjecture is equivalent to the statement that the dual graph of any simple Venn diagram has a Hamilton cycle. In this work, we construct counterexamples to Winkler's conjecture for all $n \geq 6$. As part of this proof, we computed all 3.430.404 simple Venn diagrams with $n = 6$ curves (even their number was not previously known), among which we found 72 counterexamples.

We also disprove another conjecture about the Hamiltonicity of the arrangement graph of a Venn diagram. Specifically, while working on Winkler's conjecture, Pruesse and Ruskey proved that this graph has a Hamilton cycle for every simple Venn diagram with n curves, and conjectured that this also holds for non-simple diagrams. We construct counterexamples to this conjecture for all $n \geq 4$.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Discrete mathematics

Keywords and phrases Venn diagram, Winkler's conjecture, Hamilton cycle, perfect matching, hypercube

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.22

Related Version *Full Version*: `https://arxiv.org/abs/2503.18554` [1]

Supplementary Material *Software*: `https://tmuetze.de/papers/venn.zip` [2]
archived at `swh:1:dir:baf5f66047caf6259cf73873956fbaf9305fa782`

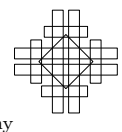
Funding Sofia Brenner, Torsten Mütze, Christian Rieck, and Francesco Verciani were supported by German Science Foundation (DFG) grant 522790373. Sofia Brenner also acknowledges funding by a postdoc fellowship of the German Academic Exchange Service (DAAD).

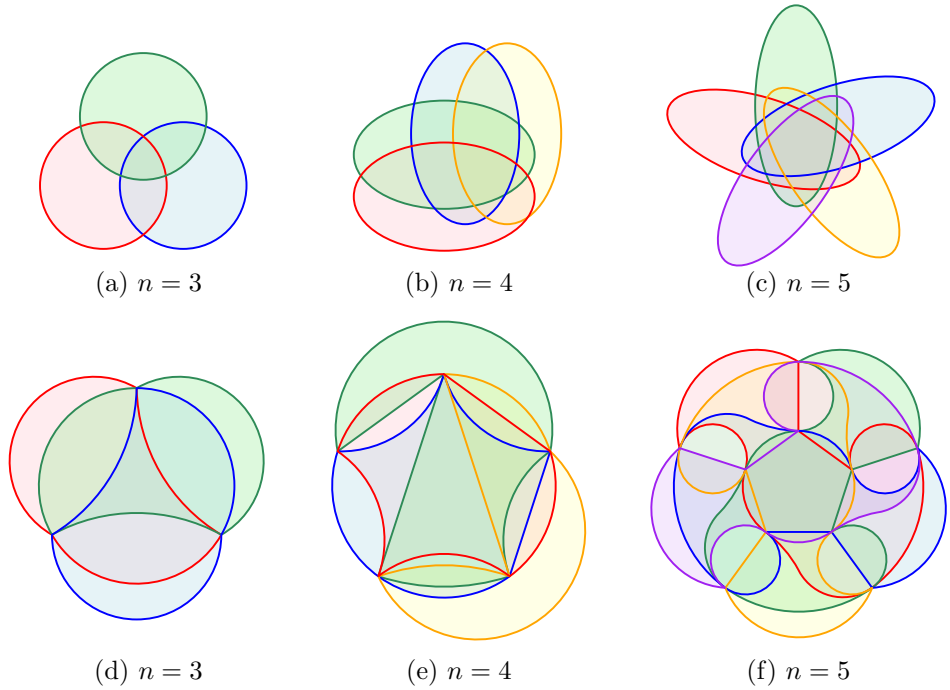
1 Introduction

Venn diagrams are a popular tool to illustrate the relations between sets and operations on them, such as union and intersection, or the well-known inclusion-exclusion principle. Historically, these diagrams were introduced by John Venn [18] (1834–1923) in the context of formal logic. Most illustrations that one finds depict the Venn diagram with three sets represented by unit circles; see Figure 1 (a). It may even come as a surprise that Venn diagrams exist for any number of sets, not just three. Formally, an *n-Venn diagram* is a collection of n simple closed curves in the plane that intersect in only finitely many points



© Sofia Brenner, Linda Kleist, Torsten Mütze, Christian Rieck, and Francesco Verciani;
licensed under Creative Commons License CC-BY 4.0





(a) $n = 3$ (b) $n = 4$ (c) $n = 5$

(d) $n = 3$ (e) $n = 4$ (f) $n = 5$

Figure 1 Venn diagrams with $n = 3, 4, 5$ curves: (a), (b), (c) are simple, whereas (d), (e), (f) are non-simple; (a), (b), (d) are reducible, whereas (c), (e), (f) are irreducible; (a), (b), (c), (d) are monotone, whereas (e), (f) are not monotone; (a), (b), (c), (d), (f) are exposed, whereas (e) is not exposed.

and create exactly 2^n regions, one for every possible combination of being inside or outside of each of the n curves. Figures 1 (b)+(c) display diagrams with $n = 4$ and $n = 5$ curves, respectively.

A Venn diagram is *simple* if at most two of the n curves intersect in any point: The three diagrams in the top row of Figure 1 are simple, whereas the three diagrams in the bottom row are non-simple. An easy application of Euler's formula shows that a simple Venn diagram has exactly $2^n - 2$ crossings, whereas non-simple diagrams may have much fewer. For humans, simple Venn diagrams tend to be easier to read than non-simple diagrams.

1.1 Winkler's conjecture

In 1984, Peter Winkler [20] raised the following intriguing conjecture about extending a simple Venn diagram by adding one curve to it.

▶ **Conjecture 1.** *Every simple n-Venn diagram can be extended to a simple $(n+1)$-Venn diagram by the addition of a suitable curve.*

He also mentioned the conjecture as an unsolved open problem in his popular book [21] on mathematical puzzles, and reiterated it in his "Puzzled" column [22] in the Communications of the ACM. The conjecture is also listed as the first open problem in Ruskey and Weston's inspiring survey [15] on Venn diagrams.

For small cases, the conjecture is easily verifiable by hand for all simple Venn diagrams on $n \in \{3, 4, 5\}$ curves, simply because there are so few of them; the 20 diagrams for $n = 5$ can be found in [6, 10]. Another class of Venn diagrams for which the conjecture is easily

seen to be true are reducible diagrams. An n-Venn diagram is *reducible* if one of its curves can be removed so that the remaining $n-1$ curves form an $(n-1)$-Venn diagram. Otherwise the diagram is called *irreducible*. The diagram in Figure 1 (c) is irreducible. If a diagram is reducible, then the curve C that can be removed splits every region of the resulting $(n-1)$-Venn diagram into two, and therefore this curve touches every region in the n-Venn diagram. We can thus add an $(n+1)$st curve by following the curve C and subdividing every region into two, changing sides along C exactly once along every segment between two consecutive crossings. Thus, the interesting instances of Conjecture 1 are irreducible diagrams.

Grünbaum [9] modified Winkler's conjecture, dropping the requirement for the diagrams to be simple. This makes the problem harder in the sense that more diagrams are allowed, but easier in the sense that there is substantially more freedom when adding the new curve. This variant of the conjecture was proved subsequently by Chilakamarri, Hamburger and Pippert [4], by using a straightforward reduction to a classical result of Whitney [19].

1.1.1 Translation to a Hamilton cycle problem

Problems on Venn diagrams such as Winkler's conjecture are best discussed by considering the *dual graph* of the Venn diagram, i.e., the plane graph that has a vertex inside every region, and edges connecting vertices corresponding to neighboring regions; see Figure 2. For this, we consider the *n-dimensional hypercube Q_n*, or *n-cube* for short, the graph formed by all bitstrings of length n, with an edge between any two bitstrings that differ in a single bit. The dual graph $Q(D)$ of a simple n-Venn diagram D satisfies the following properties:

(1) It is a connected subgraph of Q_n that is also spanning, i.e., all 2^n vertices are present. Specifically, the vertex in $Q(D)$ corresponding to a region of the diagram D is the characteristic vector of the inside/outside relation with respect to each curve, where 0 and 1 represent outside and inside, respectively.[1]

(2) It is a plane quadrangulation, i.e., it is drawn in the plane without edge crossings, and every face (including the outer face) is a 4-cycle.

(3) For every position $i \in \{1,\dots,n\}$ and every bit $b \in \{0,1\}$, the subgraph of $Q(D)$ induced by all vertices x with $x_i = b$ is connected.

Conversely, the dual of any subgraph of Q_n satisfying these properties is an n-Venn diagram.

We refer to subgraphs of Q_n satisfying (1)–(3), which are the duals of n-Venn diagrams, as *n-Venn quadrangulations*.

Extending the diagram D by a single curve means that the new curve has to enter and leave every region of D exactly once, splitting it into two, one of them inside and the other one outside the newly added curve. This corresponds to traversing a Hamilton cycle in the dual graph $Q(D)$, i.e., a cycle in the graph that visits every vertex exactly once; see Figure 2 (b)+(c).

Conjecture 1 can thus be restated equivalently as follows: For $n \geq 2$, every n-Venn quadrangulation admits a Hamilton cycle.

[1] When considering the diagram in the plane, then the inside and outside of a curve C are the bounded and unbounded region of $\mathbb{R}^2 \setminus C$, respectively.

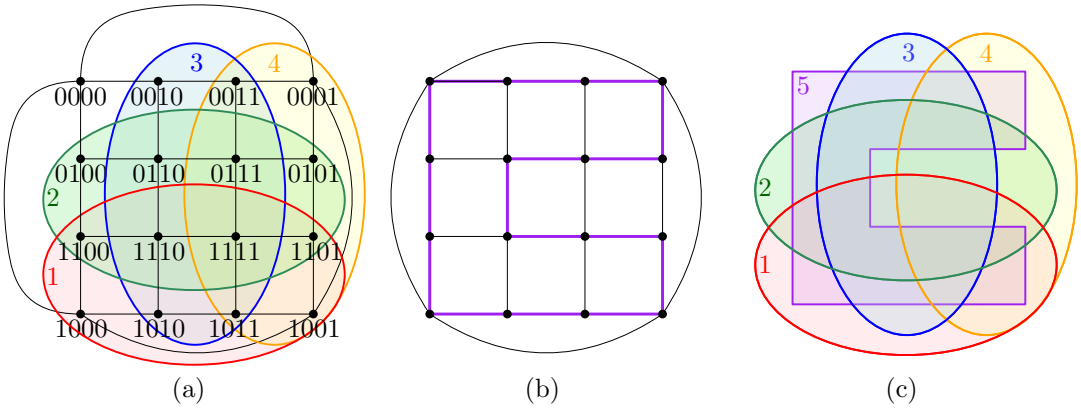


Figure 2 (a) A 4-Venn diagram and its dual graph, a 4-Venn quadrangulation; (b) a Hamilton cycle in the quadrangulation; (c) the 5-Venn diagram obtained from (a) by adding the curve corresponding to the Hamilton cycle in (b).

1.1.2 The counterexamples

As Q_n is a bipartite graph, Venn quadrangulations are bipartite, too. A necessary condition for the existence of a Hamilton cycle in a bipartite graph is the existence of a perfect matching. We provide counterexamples to Conjecture 1 by constructing, for every $n \geq 6$, an n-Venn quadrangulation that has no perfect matching and thus no Hamilton cycle. One of our counterexamples for $n = 6$ is shown in Figure 3.

▶ **Theorem 2.** *For every $n \geq 6$ there is an n-Venn quadrangulation that has no perfect matching and hence no Hamilton cycle. Its dual is a simple n-Venn diagram that cannot be extended to a simple $(n+1)$-Venn diagram by adding a suitable curve.*

As mentioned before, for $n \leq 5$ all simple n-Venn diagrams are extendable, so Theorem 2 is best possible in the sense that the counterexamples for $n = 6$ are minimal. In fact, as part of our proof of Theorem 2 we computed all distinct 6-Venn diagrams and analyzed their properties, summarized in the next theorem, Figure 4 and Table 1. Specifically, we consider two diagrams the same if they differ only by mirroring and/or stereographic projection. Equivalently, we count their dual n-Venn quadrangulations up to graph isomorphism. Prior to this work, the number of simple n-Venn diagrams was only known up to $n \leq 5$. Specifically, for each $n \leq 4$ there is only one, and for $n = 5$ there are 20 [6, 10].

To state the next theorem, we define a few relevant properties. A *k-region* in an n-Venn diagram is a region that lies inside of exactly k of the n curves (and outside of the remaining $n-k$ curves). A *monotone* Venn diagram is one in which for every $0 < k < n$, every k-region is adjacent to both a $(k-1)$-region and a $(k+1)$-region. Bultena, Grünbaum and Ruskey [3] proved that monotone diagrams are precisely the ones that can be drawn such that every curve has a convex shape. An *exposed* diagram is one in which the outer region is bounded by all n curves. It is not hard to see that any monotone diagram is also exposed. Chilakamarri, Hamburger and Pippert [5] constructed n-Venn diagrams for every n in which all regions are (combinatorial) triangles, quadrangles or pentagons, in particular, they are not exposed for $n \geq 6$.

[2] This drawing was produced using a `SageMath` program [16] developed by Manfred Scheucher for his and Felsner's work [7] on pseudocircle arrangements. We used this program with Scheucher's permission, using as input the 6-Venn quadrangulation on the right.

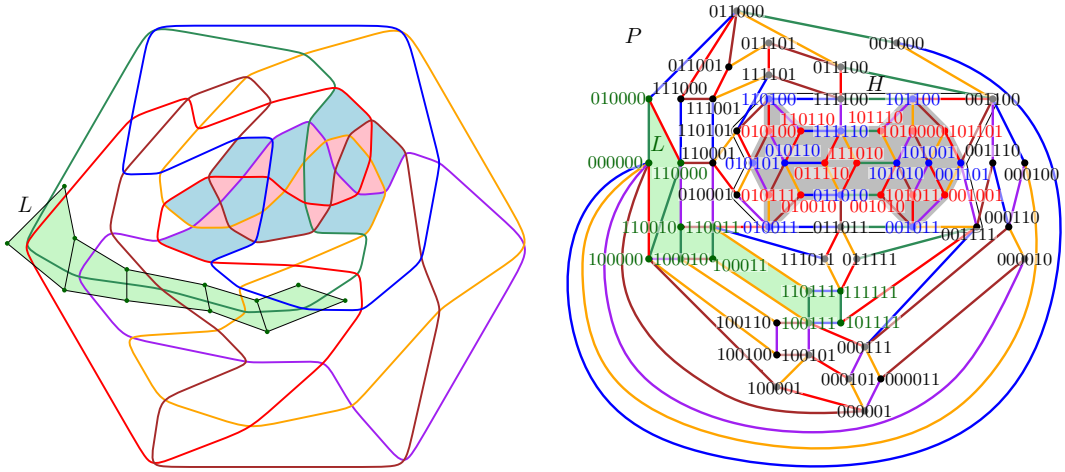


Figure 3 (Left) A counterexample to Winkler's conjecture for $n = 6$, namely a simple 6-Venn diagram that cannot be extended to a simple 7-Venn diagram by adding a suitable curve. Highlighted are 12 red regions that are surrounded by only 11 blue regions, which is a local obstruction for extendability.[2] (Right) The dual graph of the diagram on the left, namely a 6-Venn quadrangulation P that has no perfect matching and hence no Hamilton cycle. Edges are colored according to the colors of the curves. The highlighted subgraph H contains a subset of 12 red vertices in one partition class and their 11 blue neighbors, witnessing the non-existence of a perfect matching. The green subgraph L is the ladder used for extending the counterexample to all values of $n > 6$, and all those extensions contain a copy of H.

▶ **Theorem 3.** *There are* 3.430.404 *non-isomorphic 6-Venn quadrangulations (i.e., simple 6-Venn diagrams up to mirroring and/or stereographic projection). Of these,* 72 *do not have a Hamilton cycle (i.e., the diagram cannot be extended to a simple 7-Venn diagram),* 60 *do not have a Hamilton path, and* 17 *do not have a perfect matching. Furthermore, of the corresponding Venn diagrams,* 157.619 *are reducible,* 32.255 *are monotone and* 267.510 *are not exposed, and all those are extendable to a 7-Venn diagram.*

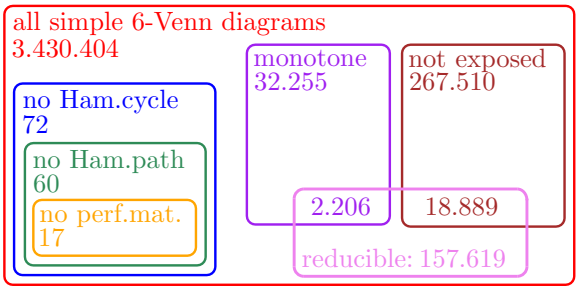


Figure 4 Census of 6-Venn diagrams and their properties stated in Theorem 3, visualized as an Euler diagram. The sizes of the boxes are not to scale.

As a consequence of Theorem 3, there are 6-Venn quadrangulations (in total $72 - 17 = 55$ many) that have a perfect matching but no Hamilton cycle, and 6-Venn quadrangulations (in total $60 - 17 = 43$ many) that have a perfect matching but no Hamilton path.

The programs we used for computing all simple 6-Venn diagrams and the result files with all 6-Venn quadrangulations are available for download [2].

1.2 Pruesse and Ruskey's conjecture

While working on Winkler's conjecture, Pruesse and Ruskey [14] considered the question of finding a Hamilton cycle not in the dual graph of a Venn diagram, but in the arrangement graph whose vertices are the crossings of the curves, and curve segments between consecutive crossings form the edges; see Figure 5. Note that these graphs may have multiple edges.

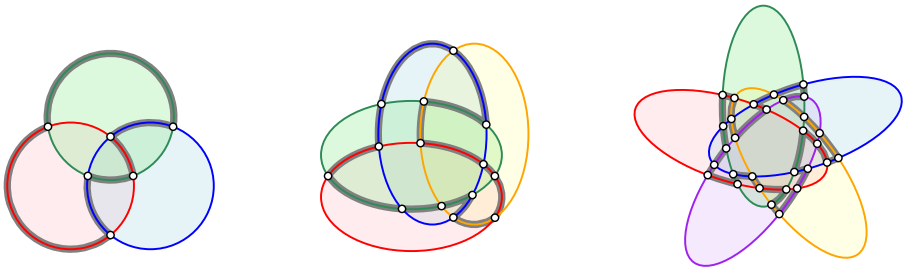


Figure 5 Hamilton cycles in the arrangement graphs of the Venn diagrams from Figure 1 (a)–(c).

▶ **Theorem 4** ([14, Cor. 1]). *For $n \geq 2$, the arrangement graph of any simple n-Venn diagram has a Hamilton cycle.*

Their proof demonstrates that the arrangement graph of a simple Venn diagram is 4-connected for $n \geq 3$, and thus a well-known theorem of Tutte [17] implies Hamiltonicity. Pruesse and Ruskey also conjectured that their result generalizes to non-simple diagrams.

▶ **Conjecture 5** ([14, Conj. 1]). *For $n \geq 2$, the arrangement graph of any (not necessarily simple) n-Venn diagram has a Hamilton cycle.*

1.2.1 The counterexamples

We disprove Conjecture 5 by providing counterexamples for all $n \geq 4$. Similarly to Theorem 2, our counterexamples have an even number of vertices and do not admit a perfect matching, which would be necessary for having a Hamilton cycle. One of the counterexamples for $n = 4$ is shown in Figure 6.

▶ **Theorem 6.** *For every $n \geq 4$, there is a non-simple n-Venn diagram whose arrangement graph has an even number of vertices and no perfect matching, and hence no Hamilton cycle.*

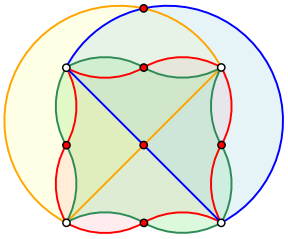


Figure 6 A counterexample to Pruesse and Ruskey's conjecture for $n = 4$, namely a non-simple 4-Venn diagram whose arrangement graph has an even number of vertices and no perfect matching, and hence no Hamilton cycle. The 6 red vertices form an independent set, and the complement has only 4 vertices, witnessing the non-existence of a perfect matching.

We remark that the counterexamples constructed in the proof of Theorem 6, including the one shown in Figure 6, are actually monotone. With straightforward modifications the diagrams can also be made *crossing*, i.e., such that whenever two curves meet, they cross.

2 Proof of Theorem 2

Our first lemma describes a technique to construct a simple $(n+1)$-Venn diagram from a simple n-Venn diagram, by carefully gluing together two slitted copies of the smaller diagram, surrounding one of the copies with an additional curve. In this process, a certain local substructure H is preserved, and we will later apply the lemma by taking for H a substructure that witnesses non-extendability. The lemma is conveniently stated in the language of the dual Venn quadrangulations.

If e is an edge of Q_n whose end vertices differ in the kth bit, then we refer to k as the *type* of the edge e. A *ladder* in an n-Venn quadrangulation is a subgraph L on pairwise distinct vertices $x_1, \dots, x_n$ and $y_1, \dots, y_n$ such that $x_1 = 0^n$, $y_n = 1^n$, for some $k \in [n] := \{1, \dots, n\}$ all pairs (x_i, y_i) for $i = 1, \dots, n$ are edges of type k in L, and for $i = 1, \dots, n-1$ all pairs (x_i, x_{i+1}) and (y_i, y_{i+1}) are edges of arbitrary type in L; see Figure 7. A ladder for $n = 6$ is highlighted in green in Figure 3. We note that in a Venn quadrangulation, every 4-cycle also bounds a face, i.e., either the inside or outside of the 4-cycle is empty, because otherwise condition ③ would be violated. Consequently, in a ladder, each of the 4-cycles $(x_i, x_{i+1}, y_{i+1}, y_i)$ for $i = 1, \dots, n-1$ bounds a 4-face.

▶ **Lemma 7.** *Consider an n-Venn quadrangulation P that contains a ladder L and a subgraph H with $L \cap H = \emptyset$. Then there is an $(n+1)$-Venn quadrangulation P' that contains a ladder L' and a copy H' of H such that $L' \cap H' = \emptyset$.*

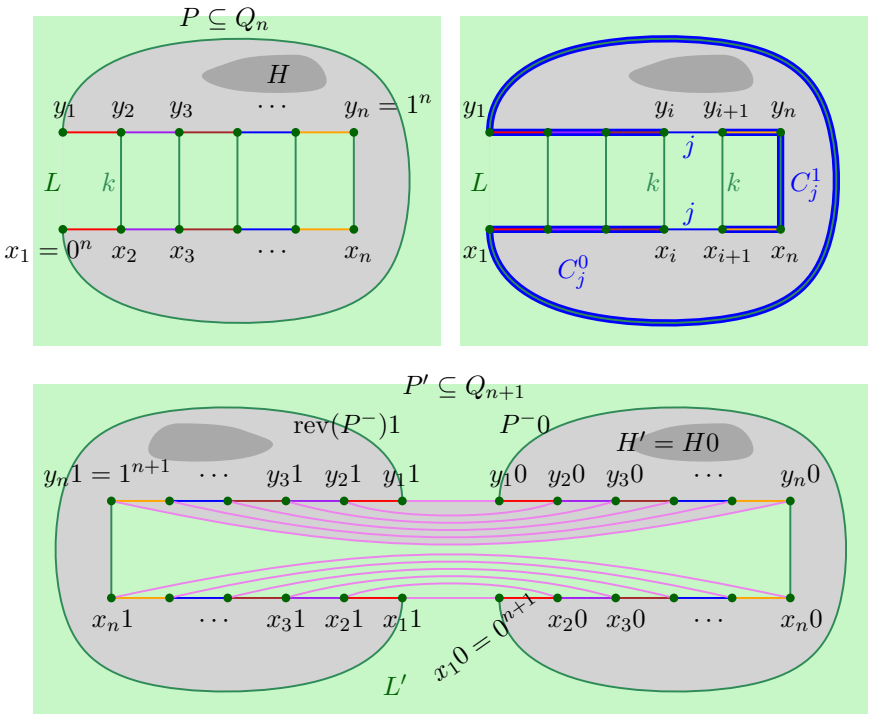


Figure 7 Illustration of Lemma 7.

The proof of Lemma 7 is illustrated in Figure 7.

Proof. Let $x_1, \dots, x_n$ and $y_1, \dots, y_n$ be the vertices of the ladder L in P as defined before, and let $k \in [n]$ be the type of the edges (x_i, y_i) for $i = 1, \dots, n$. We write C for the outer cycle of L, i.e., for the graph $C := L \setminus \{(x_i, y_i) \mid i = 2, \dots, n-1\}$. Note that in every 4-cycle of a Venn quadrangulation, there are edges of exactly two different types, and opposite edges have the same type. Consequently, both edges (x_i, x_{i+1}) and (y_i, y_{i+1}) of L (and C) have the same type for $i = 1, \dots, n-1$, and because of the condition $x_1 = 0^n$ and $y_n = 1^n$, every type $j \in [n] \setminus \{k\}$ appears for exactly one pair of opposite edges (x_i, x_{i+1}) and (y_i, y_{i+1}) for some $i = 1, \dots, n-1$. We write C_j^0 and C_j^1 for the two connected subgraphs of C obtained from C by removing both edges (x_i, x_{i+1}) and (y_i, y_{i+1}), such that the first one contains the vertex $x_1 = 0^n$ and the second one contains the vertex $y_n = 1^n$.

We assume w.l.o.g. that the 4-cycle (x_1, x_2, y_2, y_1) bounds the outer face of P. Let P^- be the graph obtained from P by removing the edges (x_i, y_i) of the ladder L for all $i = 2, \dots, n-1$. Let $\mathrm{rev}(P^-)$ denote the plane graph obtained from P^- by mirroring. Appending a 0-bit to all vertices of P^- and a 1-bit to all vertices of $\mathrm{rev}(P^-)$ gives the graphs P^-0 and $\mathrm{rev}(P^-)1$. We let P' be the plane graph obtained by connecting P^-0 and $\mathrm{rev}(P^-)1$ via $2n$ additional type-$(n+1)$ edges (x_j0, x_j1) and (y_j0, y_j1) for all $j = 1, \dots, n$. It is straightforward to check that P' satisfies conditions ①–③. In particular, the connectivity condition ③ is ensured via the subgraphs C_j^0 and C_j^1 for all bit positions $j \in [n] \setminus \{k\}$, and for the bit position k it is ensured via the subgraphs $C0$ and $C1$.

Note that P' contains a ladder L' on the vertices $x_10, \dots, x_n0, y_n0$ and $x_11, \dots, x_n1, y_n1$. Clearly, $H0$ is a copy of H in P' that is disjoint from L'. ◀

We are now ready to prove Theorem 2.

Proof of Theorem 2. The 6-Venn quadrangulation P from Figure 3 has no perfect matching, as witnessed by the marked subgraph H. Specifically, H contains a subset of 12 vertices from one partition class (marked red) whose neighborhood (marked blue) has only size 11. This is the easy direction of Hall's marriage theorem.

Repeatedly applying Lemma 7 to P with the subgraph H and the ladder L marked in Figure 3 yields an n-Venn quadrangulation without perfect matching for all $n > 6$. This completes the proof. ◀

3 Proof of Theorem 3

In this section we describe our `C++` and `SageMath` programs that compute all 6-Venn quadrangulations. For the reader's convenience, the code and data produced by them are available for download [2]. Specifically, the 6-Venn quadrangulations are provided in `graph6` format, and in a compact binary adjacency list representation. The following explanations are valid for general n, but have a reasonable computation time only for $n \leq 6$. Table 1 provides the counts of Venn diagrams with various properties for all $n = 1, \dots, 6$.

Maybe surprisingly, the computational proof of Theorem 3 was the technically most demanding part of this work, even though in the end there are relatively few 6-Venn quadrangulations in total ($< 10^7$ many). In fact, we had first found a counterexample to Winkler's conjecture for $n = 7$ (using different methods that involved SAT solvers) and a proof of Theorem 2 for $n \geq 7$, before finding the few counterexamples for $n = 6$.

Table 1 Counts of n-Venn diagrams with different properties for $n = 1, \ldots, 6$ curves. Every second row (unmarked) counts the diagrams up to mirroring and/or stereographic projection, or equivalently, the number of non-isomorphic n-Venn quadrangulations. Every other row (marked with *) counts the diagrams up to mirroring, or equivalently, the number of non-isomorphic n-Venn quadrangulations with a marked vertex which represents the outer region. We note that Mamakani, Myrvold and Ruskey [12] have counted 39.020 monotone 6-Venn diagrams before, which is in disagreement with our count of 77.395.

		n	1	2	3	4	5	6
all			1	1	1	1	20	3.430.404
	*		1	1	1	2	320	219.170.802
monotone			1	1	1	1	11	32.255
	*		1	1	1	1	18	77.395
exposed			1	1	1	1	20	3.162.894
	*		1	1	1	1	54	7.885.573
reducible			1	1	1	1	11	157.619
	*		1	1	1	2	182	10.031.770
no Hamilton cycle			0	0	0	0	0	72
	*		0	0	0	0	0	3.932
no Hamilton path			0	0	0	0	0	60
	*		0	0	0	0	0	3.192
no perfect matching			0	0	0	0	0	17
	*		0	0	0	0	0	1.088

3.1 Dynamic programming on two halves of a Venn diagram

We consider a simple n-Venn diagram D and one of its curves, say the nth one. In the dual graph $Q(D)$, the edges dual to the ones on the curve are all edges of $Q(D)$ of type n, and they form a matching M in $Q(D)$; see Figure 8 (a). Removing the matching edges M and removing the nth bit from all vertex labels splits $Q(D)$ into two subgraphs $P, P' \subseteq Q_{n-1}$, each of which satisfies the following conditions (cf. (1)–(3) from before):

(1') It is a connected subgraph of Q_{n-1} that is also spanning, i.e., all 2^{n-1} vertices are present.

(2') It is a plane graph in which every inner face is a 4-cycle, whereas the length of the outer face is at least $2n - 2$.

The outer face is bounded by a cycle C, which is the same for P and P', and we refer to it as the *boundary cycle*. Furthermore, we refer to any subgraph of Q_{n-1} satisfying (1')–(2') with boundary cycle C as a *C-quadrangulation*. Conversely, given two C-quadrangulations $P, P' \subseteq Q_{n-1}$ with $C =: (x_1, \ldots, x_\ell)$, we may consider the union

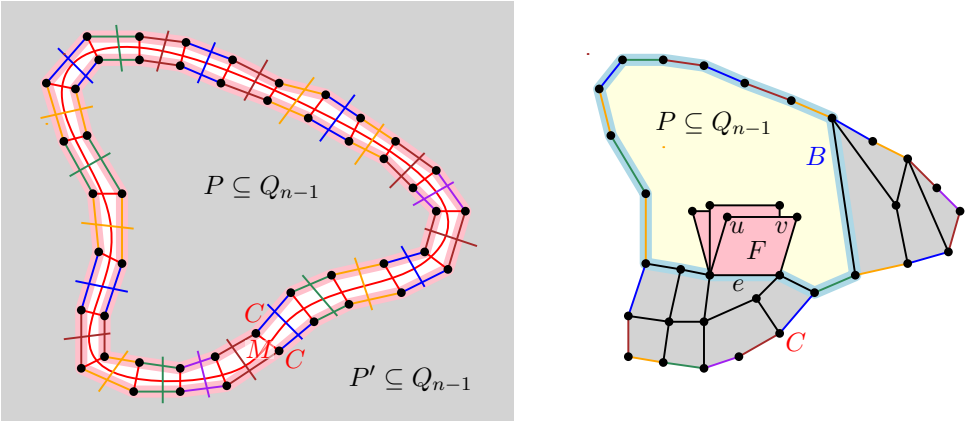
$$H := P1 \cup P'0 \cup \big\{(x_i 0, x_i 1) \mid i = 1, \ldots, \ell\big\} \subseteq Q_n, \tag{1}$$

where $P1$ and $P'0$ are obtained from P and P' by appending a 1-bit or 0-bit to all vertices, respectively. If this graph H satisfies the connectivity condition (3), then H is an n-Venn quadrangulation. In this case we say that P and P' are *compatible*.

We can therefore compute all simple n-Venn quadrangulations as follows:

(i) Compute all cycles C of Q_{n-1} up to automorphisms of the hypercube.

(ii) For every cycle C computed in step (i), compute all C-quadrangulations, i.e., subgraphs of Q_{n-1} satisfying (1')–(2') with C as the boundary cycle.



(a) (b)

Figure 8 (a) Splitting an n-Venn quadrangulation into two C-quadrangulations $P, P' \subseteq Q_{n-1}$. (b) Recursively computing C-quadrangulations by including/excluding 4-faces F to fill the inner face B.

(iii) For any two C-quadrangulations $P, P' \subseteq Q_{n-1}$ computed in step (ii), check if they are compatible, i.e., compute the graph H as in (1) and check whether H satisfies condition ③. If so, store it.

(iv) From the list of n-Venn quadrangulations stored in step (iii), filter out isomorphic duplicates.

(v) Analyze properties of all non-isomorphic n-Venn quadrangulations resulting from step (iv) according to the different properties, namely existence of a perfect matching and a Hamilton cycle or path, being monotone, exposed, reducible etc.

In the following, we describe each of these five steps in more detail. The first three steps are implemented as a single `C++` program, for efficiency reasons, and take approximately 30 hours to compute on a single desktop computer. The last two steps are implemented in `SageMath`, where several convenient high-level functions are available for working with graphs, and take approximately 60 hours to compute.

For validation purposes, we also computed the reducible 6-Venn diagrams in a different way, namely by inserting all possible Hamilton cycles into the 20 known 5-Venn quadrangulations, confirming the count of 157.619. Similarly, we also computed the monotone 6-Venn diagrams in a different way, using the method described in [12], confirming the count of 32.255.

3.1.1 Step (i): Computing all cycles C of Q_{n-1}

Any cycle $C = (x_1, \ldots, x_\ell)$ in Q_{n-1} can be described by the first vertex x_1 and the sequence of edge types $(\tau_1, \ldots, \tau_\ell)$, i.e., $\tau_i \in [n-1]$ is the type of the edge of C between vertices x_i and x_{i+1} for $i = 1, \ldots, \ell$ (where $x_{\ell+1} := x_1$). The automorphisms of the hypercube are given by all permutations of positions and taking the exclusive-or with an arbitrary fixed bitstring of length n. Consequently, we can assume w.l.o.g. that $x_1 = 0^{n-1}$. Note that for any sequence $\tau = (\tau_1, \ldots, \tau_\ell)$ of edge types from $[n-1]$, there is a unique permutation σ on $[n-1]$ such that $\sigma(\tau) = (\sigma(\tau_1), \ldots, \sigma(\tau_\ell))$ is lexicographically minimal, i.e., for every prefix $(\sigma(\tau_1), \ldots, \sigma(\tau_p))$, $p = 1, \ldots, \ell$, of the permuted sequence, if it contains r different types, then these are the types $1, \ldots, r$. Therefore, we can assume w.l.o.g. that $\tau = \sigma(\tau)$ is lexicographically minimal and furthermore, for every cyclic shift and/or reversal τ' of the sequence τ, we have $\tau \leq_{\text{lex}} \sigma(\tau')$. Factoring out the automorphisms of Q_{n-1} in this way drastically reduces the number of distinct cycles C that need to be considered.

In the following, we describe another observation that allows us to restrict the lengths of the boundary cycles to consider. As mentioned before, the total number of crossings in a simple n-Venn diagram D is $2^n - 2$. Clearly, this number equals the number of faces of $Q(D)$. The number of vertices of $Q(D)$ is 2^n, and therefore Euler's formula yields $2^{n+1} - 4$ for the number of edges of $Q(D)$. These edges split into n different matchings $M_1, M_2, \ldots, M_n$ according to their types, and we can therefore always guarantee that one matching M_i has size $|M_i| \geq \frac{2^{n+1}-4}{n}$, and the fraction on the right should be rounded up to the next even integer, because the corresponding boundary cycle in the hypercube must have even integral length. For $n = 6$ this yields the lower bound $|M_i| \geq 22$ for one of the matchings. By symmetry, we can thus restrict our search to cycles C in Q_5 of lengths $22, 24, 26, 28, 30, 32$. Note that if C has the maximum length 2^{n-1}, then it is a Hamilton cycle in Q_{n-1}, and the resulting n-Venn diagram H obtained by pairing two compatible C-quadrangulations is reducible and thus extendable.

The number of cycles of lengths $22, 24, 26, 28, 30, 32$ in Q_5 up to automorphisms is 819.049, $2.168.994, 3.859.153, 4.028.138, 1.934.320, 237.675$, respectively. The last number agrees with the fifth entry of OEIS sequence A159344, as it should.

3.1.2 Step (ii): Computing all C-quadrangulations

For a fixed cycle C in Q_{n-1} as computed in the previous step, we aim to compute all C-quadrangulations. This is done recursively, by filling the inside of C with 4-faces, one at a time. In each step, the current partial C-quadrangulation P has an inner face B that still needs to be quadrangulated; see Figure 8 (b). For this we consider every edge e along B, and every 4-cycle F in Q_{n-1} that contains e (there are $n - 3$ such 4-cycles). Every such

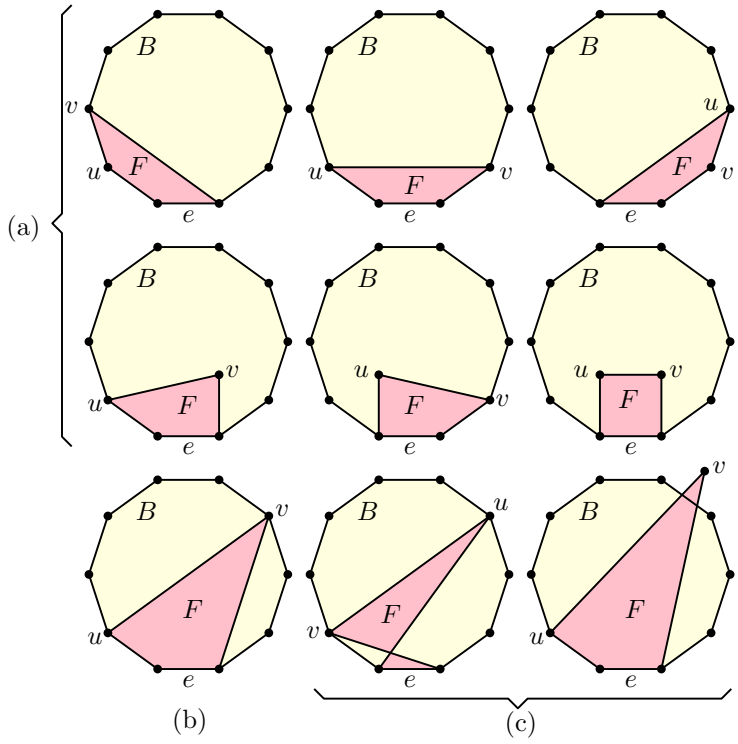


Figure 9 Different cases in the recursion to compute all C-quadrangulations.

4-cycle F containing e has two additional vertices u and v. We locate u and v in P and distinguish the following cases: If u and v are on B but in the wrong cyclic order, then F is excluded, i.e., it will not be added to P as a 4-face. Similarly, if u or v are present in P but not on B, then F is excluded. These two cases are shown in Figure 9 (c), respectively. If u and v are on B but not next to e (Figure 9 (b)), then F is ignored for this recursion step (because including F would split the inner face bounded by B into two separate faces). If u and v are next to e on B or not in P (Figure 9 (a) shows the six possible such cases), then F is eligible for a branching step. Specifically, we branch and recurse on the two possibilities of either including F, updating B accordingly, or excluding F, which does not change B. Some care has to be taken to choose a clever branching order to avoid recomputing the same C-quadrangulation along different insertion orderings of its 4-faces, and also to keep the recursion tree small, i.e., to enforce and detect dead ends quickly (for details, see our `C++` implementation [2]). Such a dead end would be an edge e on B for which all 4-cycles F containing e are excluded (apart possibly from one that can be included "behind" e).

3.1.3 Step (iii): Pairing up C-quadrangulations

For a fixed cycle C of Q_{n-1}, after computing all C-quadrangulations, we consider all pairs P, P' of them, and check if P and P' are compatible, i.e., whether the graph H defined in (1) satisfies condition ③. This is achieved by a simple graph search, run separately for every bit position $i \in [n]$ and bit value $b \in \{0, 1\}$, i.e., $2n$ searches in total. This can be slightly optimized by first doing the graph searches separately within each graph P and P', checking whether and how the subgraphs given by a fixed $i \in [n]$ and $b \in \{0, 1\}$ intersect with C (already this may lead to discarding P or P', independently of the other), and then computing the overall connectivity resulting from that.

3.1.4 Step (iv): Filtering out isomorphic graphs

After computing all n-Venn quadrangulations as described in the previous step, we need to filter out duplicates, i.e., graphs that are isomorphic copies. These isomorphic copies can arise from two different boundary cycles C and C', obtained by splitting the same diagram along two different curves. The filtering is achieved by first computing a canonical graph labeling, available in `SageMath` via the function `canonical_label()` (which internally uses the tool `bliss`), and conversion into `graph6` format, so that the isomorphism test becomes a string comparison.

3.1.5 Step (v): Analyzing properties of n-Venn quadrangulations

The different properties of a given n-Venn quadrangulation can be checked straightforwardly, either by efficient algorithms (perfect matchings, monotone, exposed, reducible) or by brute-force (Hamilton cycle/path), using standard `SageMath` functions. For $n = 6$ there are relatively few graphs and they are relatively small and sparse (64 vertices and 124 edges), so this is fast enough.

4 Proof of Theorem 6

In this section, we prove Theorem 6.

Proof of Theorem 6. We inductively construct a family of non-simple monotone n-Venn diagrams D_n for $n \geq 2$; see Figures 10 and 11. These are not the counterexamples yet, but those will be built from D_n later by small modifications. Specifically, we describe D_n as a

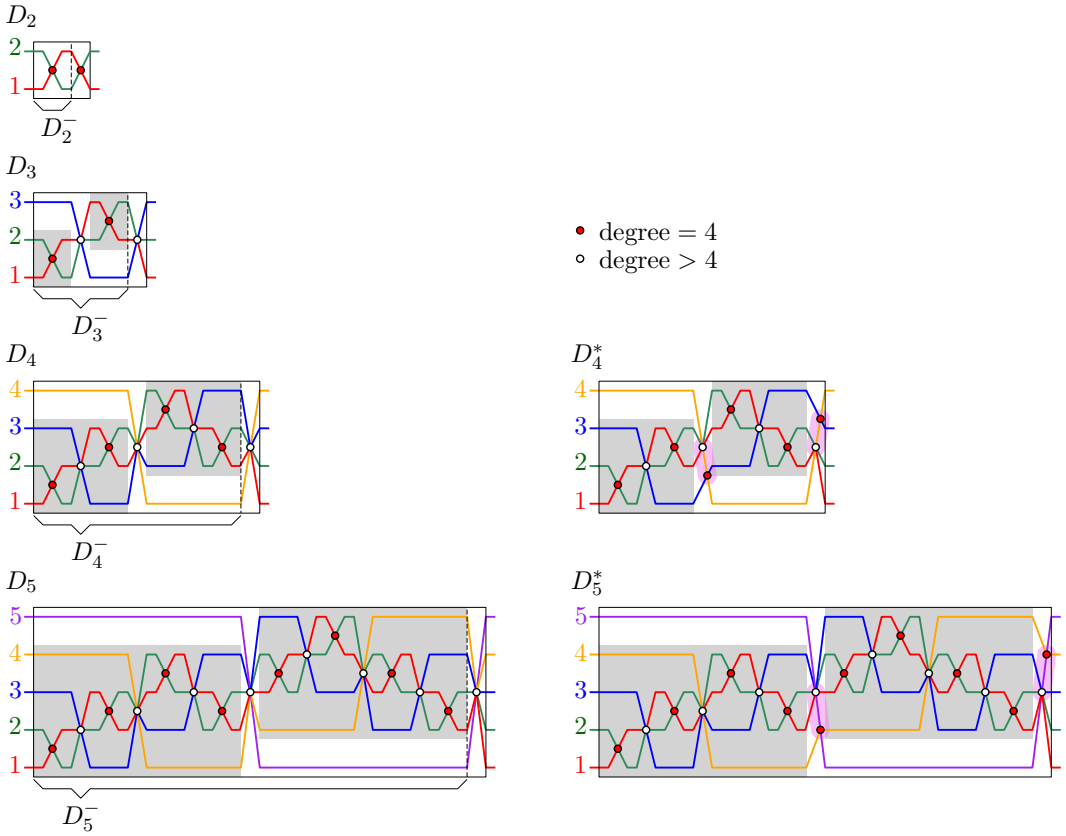


Figure 10 Non-simple Venn diagrams D_n, $n \geq 2$, and D_n^*, $n \geq 4$, constructed in the proof of Theorem 6. Another drawing of D_4^* is shown in Figure 6.

wire diagram, i.e., each crossing occurs within a horizontally separated narrow vertical strip, and outside of these strips the n curves are aligned to horizontal lines at heights $1, 2, \ldots, n$. Furthermore, the diagram wraps around cylindrically at the left and right boundary. The curves are labeled $1, 2, \ldots, n$ from bottom to top at the boundaries, according to their height.

For the induction basis $n = 2$, we take D_2 as the wire diagram obtained from the unique Venn diagram with 2 curves, which has 2 crossings; see Figure 10. For the induction step $n \to n+1$, let D_n^- be the diagram obtained from D_n by removing the rightmost crossing. To construct D_{n+1}, we take a copy of D_n^- vertically aligned at heights $1, \ldots, n$, a second copy of D_n^- but reversed (mirrored along a vertical line, i.e., left and right are interchanged) to the right of the first copy and vertically aligned at heights $2, \ldots, n+1$; see Figure 11. We then add the curve $n+1$ starting on the left at height $n+1$ above the first copy of D_n^-, staying there until its right side, then crossing below the reversed second copy of D_n^- to height 1, staying there until its right side, and finally crossing back up to height $n+1$ at the right boundary. The curves $1, \ldots, n$ are connected between the two copies of D_n^- through one additional crossing with the curve $n+1$, and to the right of the second copy through a second additional crossing with the curve $n+1$, creating two additional crossings in total, so that the ordering of curves on the right boundary becomes $1, \ldots, n+1$, as desired. One can easily verify the following properties inductively:

(i) The total number of crossings in D_n is 2^{n-1}. In particular, the arrangement graph of D_n has an even number of vertices.

(ii) The subgraph of D_n given by curve 1 and curve 2 contains all crossings, and exactly one pair of parallel edges between any two consecutive crossings. Every second crossing involves only the curves 1 and 2, i.e., the degree of the corresponding vertices is 4, and these vertices form an independent set.

(iii) Exactly two of the crossings in D_n involve all n curves, i.e., the degree of the corresponding vertices is $2n$, and the clockwise cyclic orderings of the curves around the two crossings are $n{-}1, 1, 2, \ldots, n{-}2, n, n{-}2, \ldots, 2, 1, n{-}1, n$ and $n, 1, 2, \ldots, n{-}1, n, n{-}1, \ldots, 2, 1$, respectively.

From (ii) we see in particular that the arrangement graph of D_n has a perfect matching and a Hamilton cycle, i.e., it is *not* a counterexample to Conjecture 5. However, for $n \geq 4$ we can modify D_n to obtain a diagram D_n^* as follows; see Figure 12: From the first crossing involving all n curves, we slightly shift down the curve $n-1$, and from the second crossing

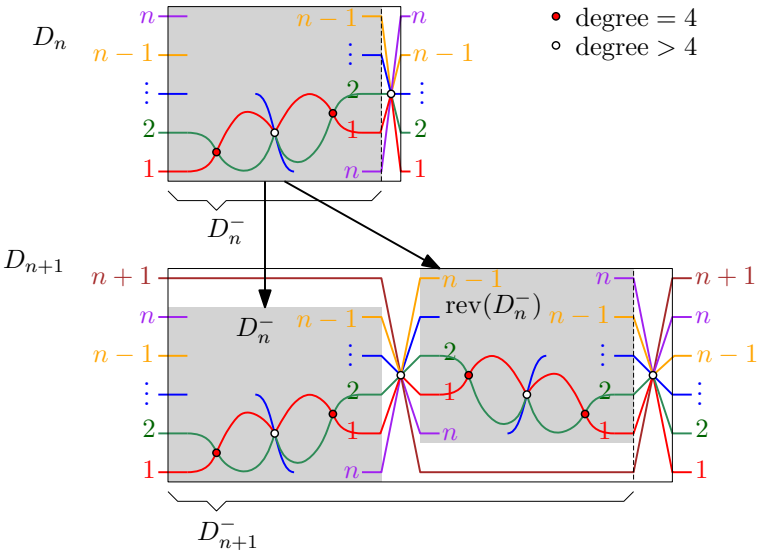


Figure 11 Illustration of the general inductive construction of D_{n+1} from D_n. The reversed copy of D_n^- is denoted rev(D_n^-).

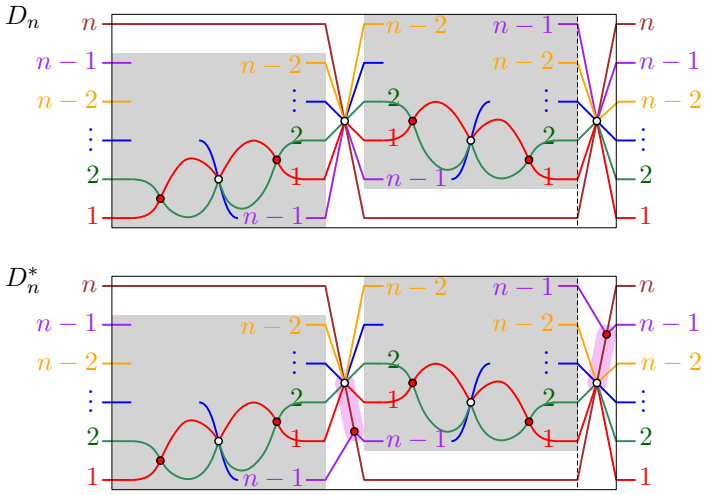


Figure 12 Modification of the diagram D_n to obtain D_n^* by slightly shifting the curve $n-1$.

involving all n curves, we slightly shift up the curve $n-1$, so that each of these two crossings is split into two, one of degree 4 and the other of degree $2n-2$ (this is possible because of (iii)). By (i), the number of vertices of D_n^* is $2^{n-1}+2$, an even number. Furthermore, in the arrangement graph of D_n^*, let U be the independent set of all degree 4 vertices and let $\overline{U}$ be its complement. From (ii) we obtain that $|U| = 2^{n-1}/2+2 = 2^{n-2}+2$ and $|\overline{U}| = 2^{n-1}/2 = 2^{n-2}$, and consequently $|U| > |\overline{U}|$. This inequality and the fact that U is an independent set shows that D_n^* does not have a perfect matching. This is the easy direction of Tutte's theorem on perfect matchings. ◀

5 Open questions

A natural question that remains open is whether Winkler's conjecture holds for monotone Venn diagrams. We believe that the answer is "no", but there are no counterexamples for $n \leq 6$ (recall Figure 4). One idea to find a monotone n-Venn quadrangulation without a perfect matching for $n \geq 7$ would be to extend the local obstruction H highlighted in Figure 3, but so far we have not succeeded in this endeavor.

Computationally, it could be interesting to find all non-simple Venn diagrams for $n = 4$ or $n = 5$ curves and analyze their properties. What can be said about the asymptotic number of n-Venn diagrams as n grows, or about the proportion of diagrams with certain properties?

In addition to these questions, there is a large number of other intriguing problems on Venn diagrams, concerning symmetry, convexity, number of crossings, restrictions on the allowed shapes of the curves (triangles, rectangles etc.), to be found in the survey [15]. For example, is there a simple n-Venn diagram with rotational symmetry made of n congruent curves for every prime number n (see [8, 11, 13])?

References

1 Sofia Brenner, Linda Kleist, Torsten Mütze, Christian Rieck, and Francesco Verciani. Disproving two conjectures on the Hamiltonicity of Venn diagrams, 2025. URL: https://arxiv.org/abs/2503.18554.

2 Sofia Brenner, Torsten Mütze, and Francesco Verciani. Venn Diagrams Program(s) and Output. Software, swhId: swh:1:dir:baf5f66047caf6259cf73873956fbaf9305fa782 (visited on 2026-05-18). URL: https://tmuetze.de/papers/venn.zip, doi:10.4230/artifacts.26092.

3 Bette Bultena, Branko Grünbaum, and Frank Ruskey. Convex drawings of intersecting families of simple closed curves. In *Canadian Conference on Computational Geometry (CCCG)*, 1999. URL: http://www.cccg.ca/proceedings/1999/c14.pdf.

4 Kiran B. Chilakamarri, Peter Hamburger, and Raymond E. Pippert. Hamilton cycles in planar graphs and venn diagrams. *J. Comb. Theory B*, 67(2):296–303, 1996. doi:10.1006/JCTB.1996.0047.

5 Kiran B. Chilakamarri, Peter Hamburger, and Raymond E. Pippert. Venn diagrams and planar graphs. *Geom. Dedicata*, 62(1):73–91, 1996. doi:10.1007/BF00240003.

6 Kiran B. Chilakamarri, Peter Hamburger, and Raymond E. Pippert. Analysis of Venn diagrams using cycles in graphs. *Geom. Dedicata*, 82(1-3):193–223, 2000. doi:10.1023/A:1005288325189.

7 Stefan Felsner and Manfred Scheucher. Arrangements of pseudocircles: Triangles and drawings. *Discret. Comput. Geom.*, 65(1):261–278, 2021. doi:10.1007/S00454-020-00173-4.

8 Jerrold R. Griggs, Charles E. Killian, and Carla D. Savage. Venn diagrams and symmetric chain decompositions in the boolean lattice. *Electron. J. Comb.*, 11(1), 2004. doi:10.37236/1755.

9 Branko Grünbaum. Venn diagrams. I. *Geombinatorics*, 1(4):5–12, 1992.

10 Peter Hamburger and Raymond E. Pippert. Simple, reducible Venn diagrams on five curves and Hamiltonian cycles. *Geom. Dedicata*, 68(3):245–262, 1997. doi:10.1023/A:1004924423245.

11 Charles E. Killian, Frank Ruskey, Carla D. Savage, and Mark Weston. Half-simple symmetric Venn diagrams. *Electron. J. Comb.*, 11(1), 2004. doi:10.37236/1839.

12 Khalegh Mamakani, Wendy Myrvold, and Frank Ruskey. Generating simple convex Venn diagrams. *J. Discrete Algorithms*, 16:270–286, 2012. doi:10.1016/j.jda.2012.04.013.

13 Khalegh Mamakani and Frank Ruskey. New roses: Simple symmetric Venn diagrams with 11 and 13 curves. *Discret. Comput. Geom.*, 52(1):71–87, 2014. doi:10.1007/S00454-014-9605-6.

14 Gara Pruesse and Frank Ruskey. All simple Venn diagrams are Hamiltonian, 2015. doi:10.48550/arXiv.1504.06651.

15 Frank Ruskey and Mark Weston. A survey of Venn diagrams. *Electron. J. Combin.*, 4(1):Dynamic Survey 5, 1997. doi:10.37236/26.

16 Manfred Scheucher. Pseudocircle arrangements tools, 2024. GitHub repository: https://github.com/manfredscheucher/pseudocircle-arrangements-tools/.

17 William T. Tutte. A theorem on planar graphs. *Trans. Amer. Math. Soc.*, 82:99–116, 1956. doi:10.2307/1992980.

18 John Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *Phil. Mag. S. 5.*, 9(59):1–18, 1880. doi:10.1080/14786448008626877.

19 Hassler Whitney. A theorem on graphs. *Ann. of Math. (2)*, 32(2):378–390, 1931. doi:10.2307/1968197.

20 Peter Winkler. Venn diagrams: Some observations and an open problem. *Congr. Numer.*, 45:267–274, 1984.

21 Peter Winkler. *Mathematical puzzles: A connoisseur's collection.* AK Peters Series, Taylor & Francis, 2004.

22 Peter Winkler. Puzzled: Where sets meet (Venn diagrams). *Commun. ACM*, 55(2):128, 2012. doi:10.1145/2076450.2076475.

Shortest Paths in Geodesic Unit-Disk Graphs

Bruce W. Brewer
Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA

Haitao Wang
Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA

Abstract

Let S be a set of n points in a polygon P with m vertices. The geodesic unit-disk graph $G(S)$ induced by S has vertex set S and contains an edge between two vertices whenever their geodesic distance in P is at most one. In the weighted version, each edge is assigned weight equal to the geodesic distance between its endpoints; in the unweighted version, every edge has weight 1. Given a source point $s \in S$, we study the problem of computing shortest paths from s to all vertices of $G(S)$. To the best of our knowledge, this problem has not been investigated previously. A naive approach constructs $G(S)$ explicitly and then applies a standard shortest path algorithm for general graphs, but this requires quadratic time in the worst case, since $G(S)$ may contain $\Omega(n^2)$ edges. In this paper, we give the first subquadratic-time algorithms for this problem. For the weighted case, when P is a simple polygon, we obtain an $O(m + n\log^3 n \log^2 m)$-time algorithm. For the unweighted case, we provide an $O(m + n\log n \log^2 m)$-time algorithm for simple polygons, and an $O(\sqrt{n}\,(n+m)\log(n+m))$-time algorithm for polygons with holes.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases unit-disk graph, geodesic distance, shortest paths, geodesic Voronoi diagrams, range emptiness queries, dynamic data structures

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.23

Related Version *Full Version*: `https://arxiv.org/abs/2603.24872`

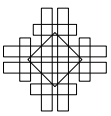
1 Introduction

Let S be a set of n points in the plane. The *unit-disk graph* induced by S has vertex set S and contains an edge between two vertices if their Euclidean distance is at most one. Equivalently, the unit-disk graph is the intersection graph of the set of congruent disks of radius 1/2 centered at the points of S: each disk corresponds to a vertex, and two disks are adjacent precisely when they intersect. Unit-disk graphs are extensively studied in computational geometry due to their rich geometric structure and numerous applications [12]. One motivation arises from wireless ad-hoc networks. Here, each point of S models a device capable of transmitting and receiving wireless signals within a fixed range, which we normalize to one. Thus, two devices can communicate directly exactly when they are connected by an edge in the corresponding unit-disk graph. Many fundamental algorithmic tasks for wireless networks reduce to classical graph problems on unit-disk graphs, such as coloring, independent set, and dominating set [12]. A central question is how to route a message between two devices s and t. If s and t are not within communication range, then the message must be relayed via intermediate devices. In graph terms, the message must follow an s-t path in the unit-disk graph, and ideally one of minimum length. For this reason, the shortest path problem in unit-disk graphs plays a key role in both theory and applications.

Another natural application of unit-disk graphs arises in motion planning with recharging/refueling. Suppose a vehicle (for example, a delivery drone) must travel from a start point s to a target point t, but lacks sufficient battery capacity to traverse this distance directly. Instead, it may need to stop at intermediate charging stations. Consider the graph



© Bruce W. Brewer and Haitao Wang;
licensed under Creative Commons License CC-BY 4.0



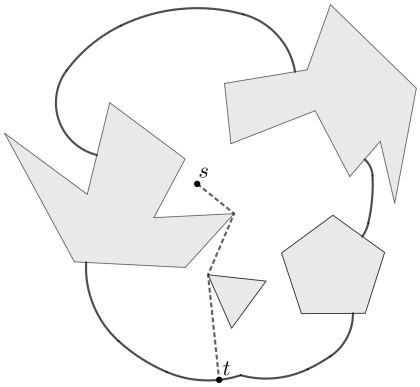


Figure 1 Illustrating a geodesic unit disk centered at s (the black solid curve is the boundary of the disk). A unit-length s-t path under geodesic distance (the dashed path) is also shown.

whose vertices represent charging stations and where two stations are connected by an edge if the vehicle can travel between them on a full charge. After scaling the maximum travel distance on a full charge to one, this graph is precisely a unit-disk graph. Thus, the motion planning problem reduces to finding a shortest s-t path in the unit-disk graph.

Now consider the above applications in the presence of obstacles that must be avoided. In wireless ad-hoc networks, an obstacle may prevent a signal from traveling directly between two devices, forcing it to bend around buildings or terrain features. In the motion planning scenario, obstacles may correspond to physical barriers or restricted regions. In such environments, Euclidean distance no longer reflects the true travel cost between two points. Instead, the relevant distance is the *geodesic distance*, i.e., the length of a shortest obstacle-avoiding path between two points. Figure 1 illustrates a "geodesic unit disk".

Problem definition. Motivated by these applications, we study *geodesic unit-disk graphs*, defined as follows. Let P be a polygonal domain with h holes and a total of m vertices in the plane. Thus, P is a closed, multiply connected region whose boundary consists of $h+1$ disjoint polygonal cycles. The holes of P, together with the region outside P, are referred to as *obstacles*. For any two points in P, their *geodesic distance* is the length of a shortest path between them in P. Let S be a set of points (also called *sites*) contained in P. The *geodesic unit-disk graph* $G(S)$ induced by S has vertex set S and contains an edge between two sites whenever their geodesic distance is at most one. In the *unweighted case*, every edge of $G(S)$ has weight 1; in the *weighted case*, the weight of an edge is the geodesic distance between its endpoints. The appropriate choice of model depends on the application. Given a source site $s \in S$, we consider the single-source shortest-path (SSSP) problem in $G(S)$: for a given polygonal domain P, site set S, and source s, compute shortest paths in $G(S)$ from s to all other vertices. To the best of our knowledge, this problem has not been studied previously.

Related work. As discussed above, problems on Euclidean unit-disk graphs (i.e., when S is given in the plane without obstacles) are well studied [12]. For the unweighted case, Cabello and Jejčič [8] gave an $O(n \log n)$-time algorithm for the SSSP problem, matching an $\Omega(n \log n)$ lower bound [8]. Chan and Skrepetos [9] later obtained an $O(n)$-time algorithm under the assumption that the points are presorted by x- and y-coordinates.

For the weighted case, Roditty and Segal [28] obtained an $O(n^{4/3+\epsilon})$-time algorithm for any $\epsilon > 0$. Cabello and Jejčič [8] improved this to $O(n^{1+\epsilon})$ time. Their runtime has since been reduced due to improvements in dynamic bichromatic closest-pair structures, first by Kaplan,

Mulzer, Roditty, Seiferth, and Sharir [20] and then by Liu [23]. Most recently, Wang and Xue [29] proposed a new algorithm that avoids dynamic bichromatic closest-pair structures, and their algorithm runs in $O(n \log^2 n)$ time; Brewer and Wang [7] further improved the time to $O(n \log^2 n / \log\log n)$ by refining a key bottleneck subproblem.

We note that various other problems involving geodesic disks in polygons have also been studied, including separators [4], clustering [5], and piercing [6], among others.

1.1 Our results

A straightforward approach is to construct $G(S)$ explicitly and then run a standard SSSP algorithm. However, $G(S)$ may contain $\Omega(n^2)$ edges, leading to quadratic time in the worst case. In this paper, we present three subquadratic-time algorithms for three problem variants.

1. For the *weighted* case, when P is a simple polygon, we obtain an $O(m + n \log^3 n \log^2 m)$-time algorithm. When applied to the Euclidean setting by taking $m = O(1)$ (i.e., P is the entire plane), our runtime is only roughly a logarithmic factor slower than the current best bound of $O(n \log^2 n / \log\log n)$ [7,29]. The Euclidean algorithms in [7,29] rely crucially on a grid decomposition of the plane, a technique that does not extend naturally to the polygon setting since a grid cell may be fragmented into multiple disconnected components by the boundary of P. We therefore use a different algorithmic strategy.
2. For the *unweighted* case in a simple polygon, we give an $O(m + n \log n \log^2 m)$-time algorithm. Applied to the Euclidean setting, this matches the $\Omega(n \log n)$ lower bound [8].
3. When P has holes, the problem becomes substantially more challenging. For the unweighted case in this setting, we develop an $O(\sqrt{n}\,(n+m)\log(n+m))$-time algorithm.

Our techniques involve solving two subproblems that may be interesting in their own rights. Indeed, solving these problems is one of the major contributions of this paper.

Subproblem 1: Deletion-only geodesic unit-disk range emptiness queries. In this problem, we preprocess a set S of n points inside a simple polygon P of m vertices to support two operations: (1) given a query point q, determine whether there exists a point $p \in S$ whose geodesic distance to q is at most 1, and if so return such a point; and (2) delete a point from S. We refer to the first operation as a *geodesic unit-disk range emptiness (GUDRE)* query.

Using the dynamic randomized nearest neighbor data structure of Agarwal, Arge, and Staals [1], each deletion can be supported in $O(\log^7 n \log m + \log^6 n \log^3 m)$ expected amortized time, and each GUDRE query can be answered in $O(\log^2 n \log^2 m)$ time. In the Euclidean case (i.e., when P is the entire plane), one can preprocess S in $O(n \log n)$ time and support both queries and deletions in $O(\log n)$ time [14,30].

We obtain the following result. After an $O(m)$-time *global* preprocessing step for P, we can, for any given set S of n points in P, build a data structure in $O(n \log^3 m + n \log^2 m \log n)$ time such that deletions take $O(\log n \log^3 m)$ amortized time and GUDRE queries take $O(\log n \log^2 m)$ time. Our result is remarkable in three aspects. First, compared to applying [1], our result improves deletion time by a factor of $O(\log^5 n)$ and improves query time by a factor of $O(\log n)$, while being deterministic rather than randomized (note that [1] also supports insertions and more general queries). Second, when applied to the Euclidean case (with $m = O(1)$), our bounds match the optimal Euclidean results of [14,30]. Third, our result has an interesting *implicit* nature: after the global preprocessing of P, the construction time for any given S depends only logarithmically on m.

Subproblem 2: Implicit additively weighted geodesic Voronoi diagram. In this problem, we wish to construct a nearest neighbor data structure for a set S of n additively weighted points in a simple polygon P with m vertices, under the requirement that the data structure be *implicitly* constructed. That is, after a global preprocessing of P, the time to build the data structure for S should depend only logarithmically on m. Such a data structure is referred to as an *implicit Voronoi diagram*.

An *explicit* diagram for S can be built in $O((n+m)\log(n+m))$ time [19], after which weighted nearest neighbor queries can be answered in $O(\log(n+m))$ time via standard point location queries [13, 21]. However, this explicit approach cannot be used in the implicit setting, as even constructing a point location data structure requires $\Omega(m)$ time.

For the unweighted version of this implicit problem, the method in [1] obtains the following: after $O(m)$-time global preprocessing, one can build a structure for S in $O(n \log n \log m + n \log^3 m)$ time that answers nearest neighbor queries in $O(\log n \log^2 m)$ time.

The weighted case is more challenging, as several geometric properties used in [1] no longer hold in the presence of weights. We develop a new method and show that, after $O(m)$-time global preprocessing of P, a data structure for S can be built in $O(n \log n \log^2 m \ + \ n \log^3 m)$ time, and each weighted nearest neighbor query can be answered in $O(\log n \log^2 m)$ time.

Outline. In the rest of the paper, after introducing preliminary concepts in Section 2, we present our algorithm for the weighted case in simple polygons in Section 3. This algorithm relies on the solutions to the two subproblems. Section 4 discusses the first subproblem. For the unweighted case, we describe our algorithm for simple polygons in Section 5, and extend it to polygons with holes in Section 6. Thus, Section 6 is the only section dealing with polygons with holes. Due to space constraints, many details and proofs are omitted but can be found in the full paper.

2 Preliminaries

We follow the notation in Sect. 1, e.g., m, n, S, P, s, and $G(S)$. Throughout the paper, P is a simple polygon unless stated otherwise (an exception is Sect. 6, where P may have holes).

We use $\overline{pq}$ to denote the line segment connecting points p and q, and $|\overline{pq}|$ to denote its (Euclidean) length. For any geometric object R in the plane, let ∂R denote its boundary.

For any two points $s, t \in P$, let $\pi(s,t)$ denote a shortest s-t path in P. If multiple shortest paths exist (which can happen only when P has holes), $\pi(s,t)$ may refer to an arbitrary one. Let $d(s,t)$ denote the geodesic distance between s and t, i.e., the length of $\pi(s,t)$. We assume that $\pi(s,t)$ is oriented from s to t. It is known that every interior vertex of $\pi(s,t)$ (i.e., except s and t) is a vertex of P. The vertex of $\pi(s,t)$ adjacent to t is called the *anchor* of t in $\pi(s,t)$, and the vertex of $\pi(s,t)$ adjacent to s is called the *anchor* of s.

Notation for the simple polygon case. When P is a simple polygon, for any three points $s, t, r \in P$, the vertex of $\pi(s,t) \cap \pi(s,r)$ farthest from s is called the *junction vertex* of $\pi(s,t)$ and $\pi(s,r)$. For a point $s \in P$ and a segment $\overline{tr} \subseteq P$, let c be the junction vertex of $\pi(s,t)$ and $\pi(s,r)$. Following standard terminology [16, 22], the paths $\pi(c,t)$ and $\pi(c,r)$ together with $\overline{tr}$ form a *funnel*, denoted $F_s(\overline{tr})$; see Figure 2. The point c is the *cusp* of the funnel, and $\pi(c,t)$ and $\pi(c,r)$ are the two *sides*, each of which is a concave chain.

When P is a simple polygon, we will frequently use the data structure of Guibas and Hershberger [15, 17], referred to as the *GH data structure*. It can be built in $O(m)$ time and supports computing $d(s,t)$ in $O(\log m)$ time for any $s, t \in P$. It can also return the anchors

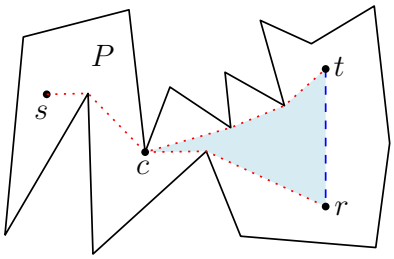


Figure 2 Illustrating a funnel $F_s(\overline{tr})$ (the light blue region).

of s and t on $\pi(s,t)$, and even a compact representation of $\pi(s,t)$ that supports binary search along the path. Furthermore, it can compute the junction vertex of $\pi(s,t)$ and $\pi(s,r)$ for any $s,t,r \in P$.

Balanced polygon decomposition (BPD). For a simple polygon P, we will use a *balanced polygon decomposition* (BPD) [11, 15, 16]. A line segment inside P connecting two vertices of P that is not an edge of P is called a *diagonal*. Chazelle [10] proved that P must contain a diagonal d that partitions P into two subpolygons, each with at most $2n/3$ vertices. Recursively decomposing the subpolygons until each becomes a triangle yields the BPD.

The BPD can be represented by a tree T_P, called the *BPD-tree*. Each node v of T_P corresponds to a subpolygon P_v of P: the root corresponds to P itself, and each leaf corresponds to a triangle. Let d_v denote the diagonal used to split P_v into its two child subpolygons. The triangles at the leaves of T_P form a triangulation of P. The entire decomposition and T_P can be constructed in $O(m)$ time [15].

Geodesic Voronoi diagram. We will also make extensive use of the *geodesic Voronoi diagram* of S in P [2, 19, 24, 27]. For a site $u \in S$, its *Voronoi region* is defined as

$$\mathrm{Vor}(u) = \{p \in P : d(u,p) \le d(v,p) \text{ for all } v \in S \setminus \{u\}\}.$$

We treat $\mathrm{Vor}(u)$ as including its boundary. For two sites $u, v \in S$, the *Voronoi edge* $\mathcal{E}(u,v)$ is the locus of points $p \in P$ satisfying $d(u,p) = d(v,p) \le d(z,p)$ for all $z \in S \setminus \{u,v\}$.

If P is a simple polygon, then $\mathcal{E}(u,v)$ has a single connected component [2], while in general polygonal domains it may have multiple components [19]. Each component consists of a sequence of hyperbolic arcs (including line segments as a degenerate case). We do not use special notation for *Voronoi vertices*, i.e., points equidistant to at least three sites and strictly farther from all others. Let $\mathcal{VD}(S)$ denote the geodesic Voronoi diagram of S in P.

The definitions extend naturally to the additively weighted case. Each site u has a weight $w(u)$, and the *weighted distance* from u to any $p \in P$ is $w(u) + d(u,p)$. Replacing geodesic distances by weighted distances yields the weighted geodesic Voronoi diagram. Note that all weights considered in this paper are additive.

3 The weighted case in a simple polygon

We now consider the SSSP problem in weighted geodesic unit-disk graphs within a simple polygon. Here, S lies inside a simple polygon P, and for every edge $e(u,v)$ of $G(S)$ connecting two sites $u, v \in S$, the weight of $e(u,v)$ is defined to be the geodesic distance $d(u,v)$.

To solve the problem, as discussed in Section 1.1, it seems difficult to extend the Euclidean case algorithm in [7, 29]. Another direction is to extend the $O(n^{1+\epsilon})$-time algorithm of Cabello and Jejčič [8], which relies on a dynamic weighted *bichromatic closest pair (BCP)*

data structure. Combining with the randomized dynamic BCP structures [20,23], the runtime improves to $O(n \log^4 n)$ expected time. But this approach does not seem to generalize to the polygon case due to the lack of an efficient dynamic weighted BCP structure under geodesic distance.

We instead take a different approach that avoids BCP altogether. Our algorithm runs in $O(m + n \log^3 n \log^2 m)$ deterministic time. Notably, when applied to the Euclidean case by treating P as the whole plane (i.e., $m = O(1)$), the running time becomes $O(n \log^3 n)$, which is even faster than the algorithm of [8] combined with dynamic BCP structures [20, 23].

In what follows, we first describe the algorithm and then discuss its implementation.

3.1 Algorithm description

Given a source point $s \in S$, our goal is to compute shortest paths from s to all other vertices v in $G(S)$. Let $d_G(v)$ denote the shortest-path distance from s to v in $G(S)$. Our algorithm maintains an array $\mathrm{dist}[\cdot]$ such that, at termination, $\mathrm{dist}[v] = d_G(v)$ for all $v \in S$. With standard modifications, the predecessor information can also be maintained so that the shortest path tree from s can be obtained.

We maintain two subsets A and B of S. The set A consists of all sites v whose values $\mathrm{dist}[v]$ have been finalized, and $B = S \setminus A$. We further partition A into two subsets, A_1 and A_2, such that no site of A_1 is adjacent to any site of B in $G(S)$. Initially, we set $\mathrm{dist}[s] = 0$, $A_2 = A = \{s\}$, $A_1 = \emptyset$, and $B = S \setminus \{s\}$. We also initialize $\mathrm{dist}[v] = \infty$ for every $v \in S \setminus \{s\}$.

In each iteration, we select $a = \arg\min_{v \in A_2} \mathrm{dist}[v]$, and perform two subroutines:

Subroutine I. Compute the set $B_a = \{\, b \in B : d(a, b) \le 1 \,\}$, i.e., all sites in B that are adjacent to a in $G(S)$.

Subroutine II. For each site $b \in B_a$, compute $\sigma_b = \arg\min_{p \in A_2} \big(\mathrm{dist}[p] + d(p, b)\big)$, and set $\mathrm{dist}[b] = \mathrm{dist}[\sigma_b] + d(\sigma_b, b)$.

Afterward, we move all sites of B_a from B into A_2, and we move a from A_2 into A_1. This completes one iteration. The algorithm terminates once A_2 becomes empty.

Algorithm correctness. Although the algorithm looks simple, its correctness is not immediate. See the full paper for the proof. In particular, we show that the algorithm maintains the following two invariants. (I) For every site $v \in A$, we have $\mathrm{dist}[v] = d_G(v)$. (II) For every site $v \in A_1$, every site u adjacent to v in $G(S)$ is already in A. Using these, we can show that after the algorithm terminates, $\mathrm{dist}[v] = d_G(v)$ holds for every site $v \in S$.

3.2 Algorithm implementation and time analysis

The main difficulty lies in implementing the two subroutines efficiently. Indeed, each subroutine corresponds precisely to one of the two subproblems introduced in Section 1.1.

We maintain the set A_2 in a min-heap with $\mathrm{dist}[v]$ as the key for all $v \in A_2$. Thus, extracting $a = \arg\min_{v \in A_2} \mathrm{dist}[v]$ takes $O(\log n)$ time, and each insertion into A_2 also takes $O(\log n)$ time. We now discuss the implementation of the two subroutines.

Subroutine I. To compute B_a, we repeatedly locate a point $b \in B$ with $d(a, b) \le 1$ and delete it from B, stopping once no such point remains. Thus, the task reduces exactly to supporting *deletion-only geodesic unit-disk range emptiness* (GUDRE) queries, as introduced in Section 1.1. Given a set B of points in P, we require a data structure that supports:

Delete: Remove a point from B.
GUDRE query: Given a point $a \in P$, determine whether there exists $b \in B$ with $d(a, b) \leq 1$, and if so, return such a point.

One of our main results in this paper is the following theorem, proved in Section 4.

▶ **Theorem 1.** *Let P be a simple polygon with m vertices, and assume that a GH data structure for P has been built in $O(m)$ time. For any set S of n points in P, we can construct a data structure for S in $O(n \log^3 m + n \log n \log^2 m)$ time such that each deletion takes $O(\log n \log^3 m)$ amortized time and each GUDRE query takes $O(\log n \log^2 m)$ time.*

Using Theorem 1, we implement Subroutine I as follows. Initially, $B = S \setminus \{s\}$. We build the GUDRE data structure for B in $O(m + n \log^3 m + n \log n \log^2 m)$ time, where the $O(m)$ term is for constructing the GH data structure. Since the algorithm performs $O(n)$ deletions and GUDRE queries overall, the total time spent in Subroutine I is $O(m + n \log n \log^3 m)$.

Subroutine II. For the second subroutine, observe that each σ_b is essentially the (additively) *weighted nearest neighbor* of b in A_2, where each site $p \in A_2$ has weight $\mathrm{dist}[p]$, and the weighted distance from p to a query point q is defined as $\mathrm{dist}[p] + d(p, q)$. Thus, we require a weighted nearest neighbor data structure for A_2.

Furthermore, since at the end of each iteration, we will move all points of B_a from B to A_2 and move a from A_2 to A_1, we need a dynamic data structure for A_2 that supports both insertions and deletions. A direct dynamic data structure supporting both insertions and deletions appears difficult to obtain. However, we show that the problem can be reduced so that only insertions are needed. The key is the following Lemma 2.

▶ **Lemma 2.** *For any site $p_1 \in A_1$ and any site $p_2 \in A_2$, we have $\mathrm{dist}[p_1] \leq \mathrm{dist}[p_2]$. Moreover, whenever a site b is added to A_2, we have $\mathrm{dist}[a] < \mathrm{dist}[b]$, and $\mathrm{dist}[p_1] < \mathrm{dist}[b]$ holds for all $p_1 \in A_1$.*

We maintain a balanced binary search tree T_A whose leaves store the sites of A in increasing order of $\mathrm{dist}[\cdot]$. By Lemma 2, the points of A can always be arranged so that all sites of A_1 appear to the left of all sites of A_2; we refer to this as the *ordering property* of T_A. For each node v of T_A, let A_v denote the set of sites whose leaves lie in the subtree rooted at v. For each v, we construct a weighted nearest neighbor data structure $\mathcal{D}_v$ for A_v, where each $p \in A_v$ has weight $\mathrm{dist}[p]$.

Given a point $b \in B$, we compute σ_b as follows. Because of the ordering property and the definition of a, all sites of $A_2 \setminus \{a\}$ lie in leaves strictly to the right of the leaf of a. In T_A, we find a set V_b of $O(\log n)$ nodes whose leaf ranges form a partition of the leaves to the right of a (including the leaf of a). For each $v \in V_b$, using $\mathcal{D}_v$, we compute a weighted nearest neighbor p_v of b in A_v and treat each p_v as a candidate for σ_b. We then return the candidate minimizing $\mathrm{dist}[p_v] + d(p_v, b)$. Thus, computing σ_b requires $O(\log n)$ weighted nearest neighbor queries plus $O(\log n)$ geodesic distance queries. Recall that a GH data structure for P can be built in $O(m)$ time, after which each distance query $d(p_v, b)$ is computable in $O(\log m)$ time.

Next we address updates. At the end of each iteration, for every $b \in B_a$, after computing $\mathrm{dist}[b]$, we insert b into T_A. This places b at a leaf, and for each node v on the path to the root, we insert b into $\mathcal{D}_v$. By Lemma 2, the ordering property of T_A is preserved automatically. Therefore, there is no need to delete a from T_A, and each $\mathcal{D}_v$ need only support insertions.

To build an insertion-only data structure $\mathcal{D}_v$, one might attempt Bentley's logarithmic method [3] together with a static geodesic Voronoi diagram for A_v. Since computing the geodesic Voronoi diagram of A_v in P requires $\Omega(m)$ time (note that the best algorithm

can compute the diagram in $O(m + |A_v| \log |A_v|)$ time [24]), rebuilding these diagrams n times would lead to $\Omega(mn)$ total time, which is too slow. Instead, we require an *implicit* construction whose dependence on m is only polylogarithmic. This is exactly the subproblem introduced in Section 1.1. In the full paper, we establish the following result.

▶ **Theorem 3.** *Let P be a simple polygon with m vertices, and assume that a GH data structure for P has been built in $O(m)$ time. Given a set S of n weighted sites in P, we can construct a weighted nearest neighbor data structure for S in $O(n \log^3 m + n \log n \log^2 m)$ time such that each query can be answered in $O(\log n \log^2 m)$ time.*

With Theorem 3, we construct $\mathcal{D}_v$ for every node v of T_A. Using Bentley's method, each insertion into $\mathcal{D}_v$ takes amortized $O(\log n \log^3 m + \log^2 n \log^2 m)$ time. Each weighted nearest neighbor query takes $O(\log^2 n \log^2 m)$ time. Since T_A has height $O(\log n)$, computing σ_b takes $O(\log^3 n \log^2 m)$ time, and inserting a site b into T_A takes amortized $O(\log^2 n \log^3 m + \log^3 n \log^2 m)$ time. Thus, the total running time of Subroutine II over the entire algorithm is $O(m + n \log^2 n \log^3 m + n \log^3 n \log^2 m)$, including the $O(m)$ time required to build the GH data structure for P in Theorem 3. It is actually bounded by $O(m + n \log^3 n \log^2 m)$.

▶ **Theorem 4.** *The SSSP problem in the weighted geodesic unit-disk graph in a simple polygon of m vertices for a set of n points can be solved in $O(m + n \log^3 n \log^2 m)$ time.*

4 Deletion-only geodesic unit-disk range emptiness (GUDRE) queries

In this section, we prove Theorem 1 for the deletion-only GUDRE query problem. The problem can be reduced to a *diagonal-separated* problem by using the balanced polygon decompositions (BPD) of simple polygons as described in Section 2.

Diagonal-separated deletion-only GUDRE queries. Consider a diagonal d that partitions P into two subpolygons P_L and P_R. In the diagonal-separated setting, we assume that all points of S lie in P_L, and every query point q lies in P_R. We have the following result.

▶ **Lemma 5.** *Let P be a simple polygon with m vertices, and assume that a GH data structure has been built for P. Given a diagonal d that partitions P into two subpolygons P_L and P_R, and a set S of n points contained in P_L, we can construct a data structure for S in $O(n \log^2 m + n \log n \log m)$ time that supports the following operations:*

1. Delete: *delete a point from S in $O(\log n \log^2 m)$ amortized time.*
2. GUDRE query: *for any query point $q \in P_R$, determine whether there exists a point $p \in S$ with $d(p, q) \leq 1$, and if so, return such a point as a witness, in $O(\log n \log m)$ time.*

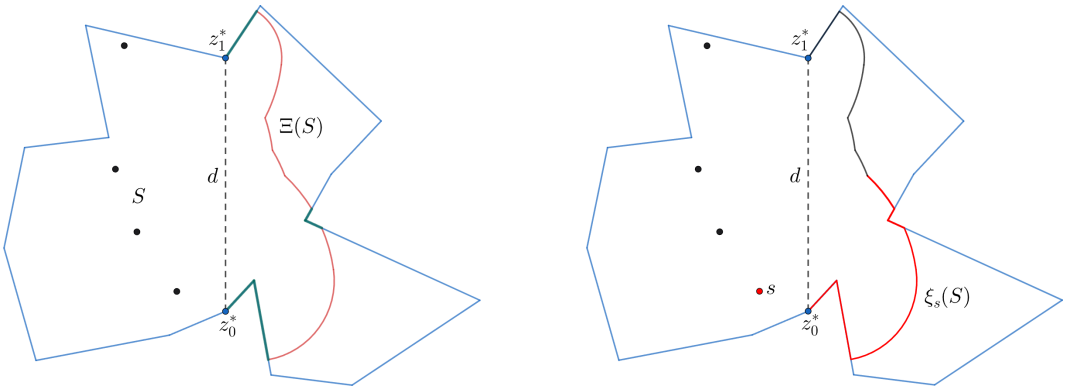
Theorem 1 can be proved using Lemma 5 and the BPD of P.

4.1 Proof of Lemma 5 – a very brief sketch

We give a very brief sketch of the proof of Lemma 5. See the full paper for all details. We follow the notation in the lemma statement, e.g., P, S, d, P_L, P_R, n, m. Without loss of generality, we assume that d is vertical and P_R is locally to the right of d. Let z_0^* and z_1^* denote the lower and upper endpoints of d, respectively.

For each point $p \in P$, we define D_p as the *geodesic unit-disk* centered at p, which consists of all points $q \in P$ with $d(p, q) \leq 1$. We define ∂P_R to be $\partial P \cap P_R$.

For any query point $q \in P_R$, we need to know whether $D_q \cap S = \emptyset$. Observe that $D_q \cap S \neq \emptyset$ if and only if $q \in \cup_{p \in S} D_p$. Based on this observation, our strategy is to maintain an implicit representation of $\mathcal{D}_R(S)$, where $\mathcal{D}_R(S)$ is the portion of $\cup_{s \in S} D_s$ in P_R. To this end, we propose a concept *right envelope* as follows.



(a) $\Xi^\circ(S)$ is red and $\Xi^\diamond(S)$ is green.

(b) $\xi_s(S)$ is colored with red.

Figure 3 Illustrations of $\Xi(S)$. The region bounded by $\Xi(S)$ and d is $\mathcal{D}_R(S)$.

Right envelope. Roughly speaking, the *right envelope* $\Xi(S)$ of $\mathcal{D}_R(S)$ is the boundary of $\mathcal{D}_R(S)$ excluding d. Specifically, $\Xi(S)$ is union of $\Xi^\circ(S)$ and $\Xi^\diamond(S)$, with $\Xi^\circ(S) = \{p \in P_R : d(S,p) = 1\}$ and $\Xi^\diamond(S) = \{p \in \partial P_R : d(S,p) < 1\}$. In general, $\Xi^\circ(S)$ is a collection of circular arcs each belonging to ∂D_s for some point $s \in S$, while $\Xi^\diamond(S)$ is a collection of segments on ∂P_R; see Figure 3a.

Note that $\mathcal{D}_R(S)$ may have multiple connected components. We can show that for each connected component $\mathcal{D}$ of $\mathcal{D}_R(S)$, $\mathcal{D} \cap d$ consists of a single segment of d and $\mathcal{D}$ is simply connected (i.e., there are no holes), meaning that $\partial\mathcal{D}$ is composed of a segment of d and a simple curve connecting the two endpoints of $\mathcal{D} \cap d$.

For any point $s \in S$, define $\xi_s(S)$ as the set of points $p \in \Xi(S)$ such that $d(s,p) \le d(s',p)$ for any other point $s' \in S$. In other words, if we consider the geodesic Voronoi diagram $\mathcal{VD}(S)$ of S, then $\xi_s(S)$ consists of all points $p \in \Xi(S)$ in the Voronoi region of s; see Figure 3b. A useful property is that if $\xi_s(S) \neq \emptyset$, then $\xi_s(S)$ is a single connected portion of $\Xi(S)$.

Answering queries using the right envelope. Intuitively, for any point $q \in P_R$, $q \in \mathcal{D}_R(S)$ if and only if q is "to the left of" $\Xi(S)$. However, the question is how to interpret "left."

In the Euclidean case where P is the entire plane and d becomes a vertical line, $\Xi(S)$ is a y-monotone curve. Due to this property, there is a trivial way to determine whether q is to the left of $\Xi(S)$: First find by binary search the arc of $\Xi(S)$ intersecting the horizontal line through q, and then determine whether q is to the left of the arc [14, 30].

In our problem, however, $\Xi(S)$ is no longer y-monotone. Intuitively, this is because P is not "large enough" and ∂P changes the "shape" of $\Xi(S)$. Thus, we need a new approach. At a very high level, our strategy is to develop some concepts analogous to those in the Euclidean case. Specifically, we introduce an order $\prec_R$ for any two points of P_R; this order corresponds to the y-coordinate order in the Euclidean case. We also define a *traversal order* $\prec_\Xi$ for any two points of $\Xi(S)$; in the Euclidean case, this would be the order of traversing on $\Xi(S)$ from bottom to top as $\Xi(S)$ is y-monotone. In addition, we define an order $\prec_L$ for all points of S based on their geodesic distances to the lower endpoint z_0^* of d (i.e., for any two points $s, t \in S$, define $s \prec_L t$ if $d(s, z_0^*) < d(t, z_0^*)$). A useful property is that these three orders are consistent with each other, which is crucial to the success of our approach.

A brief data structure description. Our data structure $\Psi(S)$ is a complete binary tree whose leaves from left to right store the points of S under the order $\prec_L$. For each node v, let S_v be the subset of points S stored in the leaves of the subtree rooted at v. We wish to store an implicit representation of $\Xi(S_v)$ at v. Let $I(S_v)$ be the set of points $s \in S_v$ such that $\xi_s(S_v) \neq \emptyset$. We consider $I(S_v)$ a list of its points sorted under $\prec_L$.

If v is the root, then we store $I(S_v)$ at v explicitly. If we store $I(S_v)$ at v explicitly for all nodes $v \in \Psi(S)$, then $\Psi(S)$ would take super-linear space. We instead employ a standard space saving technique [18, 26], where we store a point $s \in S$ in the highest node v such that $s \in I(S_v)$. Using this strategy, the total space of $\Psi(S)$ is $O(n)$.

For a node v with left child u and right child w, define $r_u = \max_{\prec_L} I(S_v) \cap I(S_u)$, i.e., among all points in $I(S_v) \cap I(S_u)$, r_u is the largest one following the order $\prec_L$. Symmetrically, define $r_w = \min_{\prec_L} I(S_v) \cap I(S_w)$. We can show that $I(S_u) \cap I(S_v)$ is a prefix of $I(S_u)$ and $I(S_w) \cap I(S_v)$ is a suffix of $I(S_w)$. Hence, $I(S_v)$ is the concatenation of $I(S_u)_{\leq r_u}$ and $I(S_w)_{\geq r_w}$, where the notation $I(S_u)_{\leq r_u}$ represents the prefix of $I(S_u)$ up to and including r_u and $I(S_w)_{\geq r_w}$ represents the suffix of $I(S_w)$ from and including r_w. We store both r_u and r_w at v. As $I(S_v)$ is the concatenation of $I(S_u)_{\leq r_u}$ and $I(S_w)_{\geq r_w}$, we only store $I(S_u)_{>r_u}$ in a list $arcs(u)$ at u and similarly store $I(S_w)_{<r_w}$ in $arcs(w)$ at w. This means that in a descent from the root of $\Psi(S)$, we can do the following. Let v be the current node in our descent and assume $arcs(v) = I(S_v)$, which is true initially when v is the root. Suppose v has left child u and right child w. We can cut $arcs(v)$ at r_u to obtain $I(S_u)_{\leq r_u}$ and $I(S_w)_{\geq r_w}$. As $I(S_u)_{>r_u}$ is stored in $arcs(u)$, we can concatenate $I(S_u)_{\leq r_u}$ and $I(S_u)_{>r_u}$ to obtain $I(S_u)$. We can do likewise to obtain $I(S_w)$. In an ascent, we can simply reverse this process to obtain $I(S_v)$ from $I(S_u)$ and $I(S_w)$ using r_u and r_w. To facilitate the queries, additional information will also be maintained such as the endpoints of $\xi_s(S_v)$ for each point $s \in arcs(v)$.

5 The unweighted simple polygon case

In this section, we present our SSSP algorithm for unweighted $G(S)$ when P is a simple polygon. To solve the problem, a natural idea is to see whether the algorithms for the Euclidean case can be adapted. Two algorithms are known that can solve the Euclidean case in $O(n \log n)$ time [8, 9]. The algorithm in [9] relies on constructing a grid in the plane, which seems inherently not applicable to the polygon setting as explained in Section 1.1.

We therefore adapt the approach of [8]. After introducing notation, we first describe the algorithm at a high level and then discuss its implementation and time analysis.

5.1 Notation

We follow the notation already defined in Sections 1 and 2, including m, n, S, P, s, $G(S)$, $\mathcal{VD}(S)$, $\mathcal{E}(u, v)$, $\mathrm{Vor}(u)$. For any subset $S' \subseteq S$ and a point $p \in P$, let $\beta_p(S')$ denote the nearest neighbor of p in S' and $d(S', p) = \min_{q \in S'} d(q, p)$. Note that $d(\beta_p(S'), p) = d(S', p)$.

We define S_i for $i = 0, 1, 2, \ldots$ to be the set of vertices of $G(S)$ whose shortest path lengths from s in $G(S)$ are equal to i. Clearly, $S_0 = \{s\}$. Define $S_{\leq i} = \cup_{j \leq i} S_j$.

Our goal is to compute the sets S_i, for $i = 0, 1, 2, \ldots$. In our discussion, we will focus on describing the algorithm to compute these sets. By slightly modifying the algorithm, we can also compute the predecessor information of the shortest paths.

Let $V(P)$ denote the set of vertices of P, which are also referred to as *obstacle vertices*. For ease of exposition, we make the following general position assumptions: (1) the shortest paths in P between any two points in $V(P) \cup S$ are unique, and (2) no point in $V(P) \cup S$ is equidistant (in geodesic distance) to two other points in $V(P) \cup S$. Note that the second

assumption implies that no obstacle vertex lies on a Voronoi edge of $\mathcal{VD}(S)$, which further means that each Voronoi edge of $\mathcal{VD}(S)$ is one-dimensional. The second assumption also implies that the nearest neighbor of a site $p \in S$ in any subset $S' \subseteq S$, i.e., $\beta_p(S')$, is unique.

We use $\mathcal{DT}(S)$ to denote the dual graph of $\mathcal{VD}(S)$. Specifically, the vertex set of $\mathcal{DT}(S)$ is S, and two sites $u, v \in S$ have an edge in $\mathcal{DT}(S)$ if their Voronoi regions in $\mathcal{VD}(S)$ share a Voronoi edge, i.e., $\mathcal{E}(u, v) \neq \emptyset$. For any site $u \in S$, we define $N_{\mathcal{DT}}(u)$ as the set of vertices of $\mathcal{DT}(S)$ adjacent to u; we also refer to sites of $N_{\mathcal{DT}}(u)$ as *neighbors* of u in $\mathcal{DT}(S)$. For a subset $S' \subseteq S$, by slightly abusing the notation, we let $N_{\mathcal{DT}}(S') = \bigcup_{u \in S'} N_{\mathcal{DT}}(u)$.

5.2 Algorithm description

We first construct the geodesic Voronoi diagram $\mathcal{VD}(S)$. The algorithm then proceeds in iterations. Initially, we have $S_0 = \{s\}$. In the i-th iteration for $i \geq 1$, we compute the set S_i as follows. We assume that S_{i-1} is available. Let $Q = N_{\mathcal{DT}}(S_{i-1}) \setminus S_{\leq i-1}$, i.e., the sites in S adjacent to those of S_{i-1} in $\mathcal{DT}(S)$ but not in $S_{\leq i-1}$. For each site $p \in Q$, we remove p from Q and determine whether $d(S_{i-1}, p) \leq 1$. If so, we add p to S_i. In addition, we update $Q = Q \cup (N_{\mathcal{DT}}(p) \setminus S_{\leq i})$, i.e., add the sites of $N_{\mathcal{DT}}(p)$ that are not in $Q \cup S_{\leq i}$ to Q for the current S_i. We repeat this until Q becomes empty. The i-th iteration finishes once Q become empty. If $S_i = \emptyset$, then the algorithm stops; otherwise, we proceed with the next iteration. See Algorithm 1 for the pseudocode.

Correctness. It is not obvious why the algorithm is correct. See the full paper for the proof.

Algorithm 1 SSSP algorithm for the simple polygon case.

```
1  i ← 0, S_0 ← {s}, build VD(S) and DT(S);
2  while S_i ≠ ∅ do
3  |  i ← i + 1, S_i ← ∅;
4  |  Q ← N_DT(S_{i-1}) \ S_{≤i-1} ;
5  |  while Q ≠ ∅ do
6  |  |  p ← pop(Q);
7  |  |  if d(S_{i-1}, p) ≤ 1 then
8  |  |  |  S_i ← S_i ∪ {p};
9  |  |  |  Q ← Q ∪ (N_DT(p) \ S_{≤i}) ;
10 return S_i for i = 0, 1, ...;
```

5.3 Algorithm implementation and time analysis

First of all, computing $\mathcal{VD}(S)$ can be done in $O(m + n \log n)$ time [24] (see also [2, 25, 27] for other algorithms). Note that the combinatorial size of $\mathcal{VD}(S)$ is $O(n + m)$ [2]. Therefore, $\mathcal{DT}(S)$ can be obtained from $\mathcal{VD}(S)$ in additional linear time. In addition, since $\mathcal{DT}(S)$ has n vertices and is a planar graph, $\mathcal{DT}(S)$ has $O(n)$ edges.

Throughout the whole algorithm, each edge of $\mathcal{DT}(S)$ is considered at most four times (twice for each vertex) in the construction of the neighborhood sets. Specifically, the edge (u, v) for $u \in S_i$ will be considered by Line 9 in the i-th iteration and by Line 4 in the $i+1$-th iteration. There are $O(n)$ edges in $\mathcal{DT}(S)$, so there are at most $O(n)$ insertions (and thus pop operations) to Q. Hence, the while loop on Line 5 has at most $O(n)$ iterations throughout the whole algorithm. It remains to determine the time complexity for the operation of checking whether $d(S_{i-1}, p) \leq 1$ on Line 7.

Notice that $d(S_{i-1}, p) \leq 1$ if and only if S_{i-1} has a point q with $d(q, p) \leq 1$. Hence, the problem can be solved by a geodesic unit-disk range emptiness (GUDRE) query. Using Theorem 1, we construct a GUDRE data structure for S_{i-1} in $O(|S_{i-1}| \cdot (\log^3 m + \log n \log^2 m))$ time at the beginning of each i-th iteration of Algorithm 1. Using a GUDRE query, whether $d(S_{i-1}, p) \leq 1$ can be determined in $O(\log n \log^2 m)$ time by Theorem 1. Note that no deletions on S_{i-1} are needed. Since $\sum_i |S_{i-1}| = n$, the overall time for constructing the GUDRE data structure for S_{i-1} in all iterations of the algorithm is $O(n(\log^3 m + \log n \log^2 m))$.

We conclude that the total time of the algorithm is $O(m + n \log n + n(\log^3 m + \log n \log^2 m))$, which is $O(m + n \log n \log^2 m)$.

▶ **Theorem 6.** *Given a simple polygon P of m vertices, a set S of n points in P, and a source point $s \in S$, we can compute shortest paths from s to all points of S in the geodesic unit-disk graph of S in $O(m + n \log n \log^2 m)$ time.*

6 The unweighted polygonal domain case

When P is a polygonal domain with holes, the framework of Algorithm 1 continues to apply. The main difficulty is algorithmic. Unlike the simple polygon setting, we do not have an efficient analogue of Theorem 1 for polygonal domains (even without deletions) that would allow us to evaluate the condition $d(S_{i-1}, p) \leq 1$ on Line 7.

As before, the algorithm proceeds in iterations, with the ith iteration responsible for computing S_i. Our key idea is to distinguish between iterations based on the size of S_i. If $|S_i| \geq \sqrt{n}$, we classify the iteration as *heavy*; otherwise, it is *light*. We design separate procedures for these two types, referred to as the *heavy-iteration procedure* and the *light-iteration procedure*, respectively. At the start of iteration i, the value of $|S_i|$ is not known in advance. Our strategy is therefore to always begin with the light-iteration procedure. If, during its execution, more than $\sqrt{n}$ sites are discovered for S_i, we immediately abort the light-iteration procedure and switch to the heavy-iteration procedure.

A useful observation is that there can be at most $\sqrt{n}$ heavy iterations, since the sets $S_1, S_2, \ldots$ are disjoint, their union is S, and each heavy iteration has at least $\sqrt{n}$ sites.

We only give a very brief sketch below. See the full paper for the details. We follow the notation and the discussion in Section 5.1. While the heavy-iteration procedure is relatively easy, the light procedure is quite interesting.

Heavy-iteration procedure. We compute the geodesic Voronoi diagram $\mathcal{VD}(S_{i-1})$ of S_{i-1} in P in $O((|S_{i-1}| + m) \log(|S_{i-1}| + m))$ time. With the diagram and a point location data structure, for each site $p \in S \setminus S_{\leq i-1}$, we can compute $\beta_p(S_{i-1})$ and its geodesic distance from p in $O(\log(m + |S_{i-1}|))$ time [19]; note that their geodesic distance is $d(S_{i-1}, p)$. If $d(S_{i-1}, p) \leq 1$, we add p to S_i. This computes S_i in $O((|S_i| + m) \log(|S_{i-1}| + m) + n \log(|S_{i-1}| + m))$ time, which is $O((n + m) \log(n + m))$. Since the number of heavy iterations in the entire algorithm is $O(\sqrt{n})$, the total time of all heavy procedures in the algorithm is $O(\sqrt{n}(n + m) \log(n + m))$.

Light-iteration procedure. The procedure runs in rounds, and each round computes at least one new point for S_i. If a round does not compute any new point for S_i, then we can assert that all sites of S_i have been computed. If after a round $|S_i|$ becomes larger than or equal to $\sqrt{n}$, then we terminate the procedure and switch to the heavy-iteration procedure.

In each round, we search in the neighborhoods of the sites of S_{i-1} and the current S_i. We find that we do not need $\mathcal{VD}(S_{i-1})$ in the entire P to conduct this search, and only a "small portion" of it suffices. More specifically, suppose there is a shortest path $\pi(S_{i-1}, v)$ from a

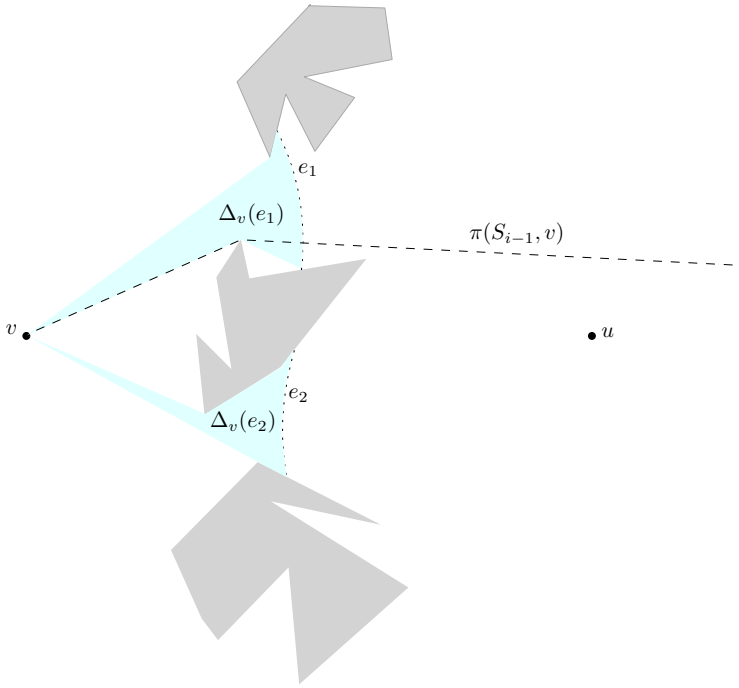


Figure 4 The gray polygons are obstacles. The path $\pi(S_{i-1}, v)$ (dashed polyline) passes through the connected component e_1 (dotted curve) of $\mathcal{E}(u, v)$, so $\pi(S_{i-1}, v) \cap \mathrm{Vor}(v)$ is contained in $\Delta_v(e_1)$ (the upper light blue region).

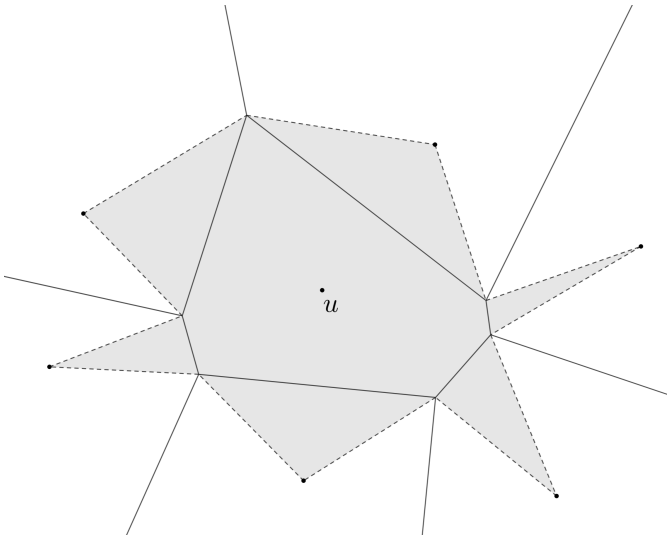


Figure 5 The Voronoi diagram is drawn with solid segments. The shaded region is $\bigstar(u)$.

site of S_{i-1} to v and $\pi(S_{i-1}, v)$ enters $\mathrm{Vor}(v)$ through $\mathrm{Vor}(u)$ for a neighbor $u \in N_{\mathcal{DT}}(v)$. Consider the Voronoi edge $\mathcal{E}(u, v)$ of $\mathcal{VD}(S)$ between u and v. This edge may be comprised of multiple connected components $e_1, \ldots, e_k$ where each component is a sequence of hyperbolic arcs. For each component e, let $\Delta_v(e)$ be the subregion of $\mathrm{Vor}(v)$ bounded by $\pi(v, a)$, $\pi(v, b)$, and e, where a and b are the two endpoints of e (note that $\pi(v, a)$ and $\pi(v, b)$ must be in $\mathrm{Vor}(v)$). We observe that for any point p in e, there exists a shortest path $\pi(p, v)$ that is inside $\Delta_v(e)$. Therefore, if we want to include the shortest path $\pi(S_{i-1}, v)$ through $\mathcal{E}(u, v)$, the only portion of $\mathrm{Vor}(v)$ that we need to include when constructing the Voronoi diagram for S_{i-1} is union of $\Delta_v(e)$ for all components $e \subseteq \mathcal{E}(u, v)$. See Figure 4. Motivated by this, for each site u, we define $\bigstar(u)$ to be the union of $\mathrm{Vor}(u)$ and all $\Delta_v(e)$ for all $v \in N_{\mathcal{DT}}(u)$ and all connected components $e \subseteq \mathcal{E}(u, v)$; see Figure 5. By including $\bigstar(u)$ in our "work region" for computing the Voronoi diagram for S_{i-1}, we ensure that for all $v \in N_{\mathcal{DT}}(u)$, if $\pi(S_{i-1}, v)$ crosses $\mathcal{E}(u, v)$, then this path will be included in our work region. This leads to a total of $O(\sqrt{n}(n + m) \log(n + m))$ time for all light procedures.

References

1. Pankaj K. Agarwal, Lars Arge, and Frank Staals. Improved Dynamic Geodesic Nearest Neighbor Searching in a Simple Polygon. In *Proceedings of the 34th International Symposium on Computational Geometry (SoCG)*, pages 4:1–4:14, 2018. doi:10.4230/LIPIcs.SoCG.2018.4.
2. Boris Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4:109–140, 1989. doi:10.1007/BF01553882.
3. Jon L. Bentley. Decomposable searching problems. *Information Processing Letters*, 8:244–251, 1979. doi:10.1016/0020-0190(79)90117-0.
4. Mark de Berg, Sándor Kisfaludi-Bak, Morteza Monemizadeh, and Leonidas Theocharous. Clique-based separators for geometric intersection graphs. *Algorithmica*, 85:1652–1678, 2022. doi:10.1007/s00453-022-01041-8.
5. Magdalene G. Borgelt, Marc van Kreveld, and Jun Luo. Geodesic disks and clustering in a simple polygon. In *Proceedings of the 18th International Conference on Algorithms and Computation (ISAAC)*, pages 656–667, 2007. doi:10.5555/1781574.1781647.
6. Prosenjit Bose, Paz Carmi, and Thomas C. Shermer. Piercing pairwise intersecting geodesic disks. *Computational Geometry: Theory and Applications*, 98(101774), 2021. doi:10.1016/j.comgeo.2021.101774.
7. Bruce W. Brewer and Haitao Wang. An improved algorithm for shortest paths in weighted unit-disk graphs. In *Proceedings of the 36th Canadian Conference on Computational Geometry (CCCG)*, pages 57–64, 2024. arXiv:2407.03176.
8. Sergio Cabello and Miha Ejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry: Theory and Applications*, 48:360–367, 2015. doi:10.1016/j.comgeo.2014.12.003.
9. Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC)*, pages 24:1–24:13, 2016. doi:10.4230/LIPIcs.ISAAC.2016.24.
10. Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9(2):145–158, 1993. doi:10.1007/BF02189314.
11. Bernard Chazelle and Leonidas J. Guibas. Visibility and intersection problems in plane geometry. *Discrete and Computational Geometry*, 4:551–589, 1989. doi:10.1007/BF02187747.
12. Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990. doi:10.1016/0012-365X(90)90358-O.
13. Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986. doi:10.1137/0215023.

14 Alon Efrat, A. Itai, and Matthew J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001. doi:10.1007/s00453-001-0016-8.

15 Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39:126–152, 1989. doi:10.1016/0022-0000(89)90041-X.

16 Leonidas J. Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987. doi:10.1007/BF01840360.

17 John Hershberger. A new data structure for shortest path queries in a simple polygon. *Information Processing Letters*, 38(5):231–235, 1991. doi:10.1016/0020-0190(91)90064-O.

18 John Hershberger and Subhash Suri. Applications of a semi-dynamic convex hull algorithm. *BIT*, 32:249–267, 1992. doi:10.1007/BF01994880.

19 John Hershberger and Subhash Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999. doi:10.1137/S0097539795289604.

20 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discrete and Computational Geometry*, 64:838–904, 2020. doi:10.1007/s00454-020-00243-7.

21 David G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12:28–35, 1983. doi:10.1137/0212002.

22 D.T. Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984. doi:10.1002/net.3230140304.

23 Chih-Hung Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. *SIAM Journal on Computing*, 51:723–765, 2022. doi:10.1137/20M1388371.

24 Eunjin Oh. Optimal algorithm for geodesic nearest-point Voronoi diagrams in simple polygons. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 391–409, 2019. doi:10.1137/1.9781611975482.25.

25 Eunjin Oh and Hee-Kap Ahn. Voronoi diagrams for a moderate-sized point-set in a simple polygon. *Discrete and Computational Geometry*, 63:418–454, 2020. doi:10.1007/s00454-019-00063-4.

26 Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2):166–204, 1981. doi:10.1016/0022-0000(81)90012-X.

27 Evanthia Papadopoulou and D.T. Lee. A new approach for the geodesic Voronoi diagram of points in a simple polygon and other restricted polygonal domains. *Algorithmica*, 20:319–352, 1998. doi:10.1007/PL00009199.

28 Liam Roditty and Michael Segal. On bounded leg shortest paths problems. *Algorithmica*, 59:583–600, 2011. doi:10.1007/s00453-009-9322-3.

29 Haitao Wang and Jie Xue. Near-optimal algorithms for shortest paths in weighted unit-disk graphs. *Discrete and Computational Geometry*, 64:1141–1166, 2020. doi:10.1007/s00454-020-00219-7.

30 Haitao Wang and Yiming Zhao. Computing the minimum bottleneck moving spanning tree. *Algorithmica*, 22:6:1–6:22, 2026. doi:10.1007/s00453-025-01358-0.

Single-Criteria Metric r-Dominating Set Problem via Minor-Preserving Support

Reilly Browne
Dartmouth College, Hanover, NH, USA

Hsien-Chih Chang
Dartmouth College, Hanover, NH, USA

Abstract

Given an unweighted graph G, the *minimum r-dominating set problem* asks for a subset of vertices S of the smallest cardinality, such that every vertex in G is within radius r to some vertex in S. While the r-dominating set problem on planar graph admits PTAS from Baker's shifting/layering technique when r is a constant, the problem becomes significantly harder when r can depend on n. In fact, under Exponential-Time Hypothesis, Fox-Epstein *et al.* [SODA 2019] observed that no efficient PTAS can exist for the unbounded r-dominating set problem on planar graphs. One may consider even harder weighted-variant known as the *vertex-weighted metric r-dominating set*, where edges are associated with lengths, and every vertex is associated with a positive-valued weight, and the goal is to compute an r-dominating set with minimum total weight. As a result, people resorted to *bicriteria* algorithms by allowing the returned solution to use radius-$(1+\varepsilon)r$ balls instead, in addition to the total weight being a $1+\varepsilon$ approximation to the optimal value.

We establish the first *single-criteria* polynomial-time $O(1)$-approximation algorithm for the vertex-weighted metric r-dominating set problem on planar graphs when r is part of the input, and can be arbitrarily large compared to n. Our new (single-criteria) $O(1)$-approximation algorithm uses the quasi-uniformity sampling technique of Chan *et al.* [SODA 2012] by bounding the *shallow cell complexity* of the (unbounded) radius-r ball system to be linear in n. To this end we have two technical innovations:

1. The discrete ball system on planar graphs are neither pseudodisks nor have well-defined boundaries for standard union-complexity arguments. We construct a *support graph* for arbitrary distance ball systems as contractions of Voronoi cells; the sparseness comes as a byproduct.

2. We present an assignment of each depth-(≥ 3) cell to a unique 3-tuple of ball centers. This allows us to use standard Clarkson-Shor techniques to reduce the counting to cells of depth *exactly* 3, which we prove to be size $O(n)$ by a novel geometric argument based on our support being a Voronoi contraction.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Graph theory

Keywords and phrases Minimum dominating set, planar graphs, shallow cell complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.24

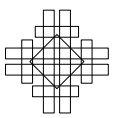
Related Version *Full Version*: https://doi.org/10.5281/zenodo.19271844

Funding *Reilly Browne*: Supported by the U.S. National Science Foundation CAREER Award under the Grant No. CCF-2443017.
Hsien-Chih Chang: Supported by the U.S. National Science Foundation CAREER Award under the Grant No. CCF-2443017.



© Reilly Browne and Hsien-Chih Chang;
licensed under Creative Commons License CC-BY 4.0





1 Introduction

The concept of *distance balls* is of fundamental importance to any given metric space (X, d) as the concept of *open sets* to any topological space. A *distance ball* $B(v, r)$ in a metric space (X, d) with radius r centered at point v is the subset of points $\{u \in X : d(u, v) \leq r\}$ in X. Many of the algorithmic success on graphs can be attributed to new understandings of the structure of distance balls in the corresponding graph metric.

The role of distance balls. Take the *minimum dominating set* – one of the 21 famous NP-complete problems of Karp – as an example. Given an unweighted graph G, the *minimum dominating set problem* (Dominating Set for short) asks for a subset of vertices S of smallest cardinality, such that every vertex in G is either in S, or is adjacent to some vertex in S. Dominating Set can also be interpreted as an instance of the *set cover problem*: Consider the set $\mathcal{R} \coloneqq \{B(v, 1) : v \in V_G\}$ of 1-neighborhood balls. Observe that a set S is dominating in G if and only if every vertex is covered by the subset of 1-neighborhood balls in $\mathcal{R}$ centered in S. In other words, a minimum solution of the set-cover instance $(V_G, \mathcal{R})$ is a minimum dominating set in G. This simple observation immediately implies an $O(\log n)$-approximation to Dominating Set, where n is the number of nodes in G, by the standard greedy set cover algorithm. $O(\log n)$-approximation is the best possible (under popular beliefs in complexity) if we make no further assumptions on the graph [25, 15]; this is essentially because the interaction between 1-neighborhood balls in general graphs can be complex and unruly.

What if we consider special classes of graphs with additional geometry constraints? One of the most well-studied examples is *planar graphs*, graphs with a drawing in the plane where no two edges cross each other. Planar graphs exhibit many helpful traits for algorithm design; particularly, several partitioning tools exist, including *separators* [43, 47, 59], *low-diameter decompositions* [41, 4, 9], and an algorithmic paradigm now famously known as *Baker's shifting/layering technique* [1] which utilizes diameter-treewidth property [23] and the decomposability of planar graphs into k-outerplanar graphs. The idea is to compute a BFS-tree from an arbitrary node, then focus only on a few consecutive layers at a time. If the optimization problem at hand is *local* (for Dominating Set, node coverage can be checked by examining immediate neighbors), then a major portion of the optimal solution would be preserved within the consecutive layers. By iterating over the initial shift value and removing one out of every $O(1/\varepsilon)$ layers in the BFS-tree, one may guarantee a $(1 + \varepsilon)$-approximation to the optimal solution. This is the main idea behind the polynomial-time approximation scheme (PTAS) of Dominating Set on unweighted planar graphs [1].

Not surprisingly, the same layering technique can be applied to the *r-dominating set problem* (r-Dominating Set for short) in planar graphs for any absolute constant r, where the goal is similar to Dominating Set but every node in S can "cover" a vertex subset of radius r [18]. r-Dominating Set was introduced by Barilan, Kortsarz, and Peleg [3] and later rediscovered in several other context [30, 18]. However, one encounters some trouble when generalizing Baker's technique to minor-free graphs. This is because arbitrary minor-free graph no longer satisfies the diameter-treewidth property (imagine adding one extra vertex connecting to every node in a planar graph; the diameter becomes 2 but the graph is K_6-minor-free). One possible way to overcome the issue is to apply the Robertson-Seymour structural theorem for minor-free graphs [33, 58], which seems to be overkill. Another, possibly more satisfying solution, is to understand what the $O(1)$-radius neighborhood balls can look like in a minor-free graph. This is the route taken by the *local search* approach. On planar graphs, typical local search analysis [10] is usually done by applying an *exchange*

argument on the base graph G, utilizing the fact that planar graphs have small separators [43] (in the form of an r-division [29]). Har-Peled and Quanrud [35] observed that, for problems like r-dominating set, the exchange argument in a typical local search analysis can be performed on the *intersection graph* of $O(1)$-neighborhood balls [48, 12] (more accurately, the subgraph $G_{\mathcal{R}}$ induced by the union of the local search solution and the optimal solution). They further observed that the intersection graph $G_{\mathcal{R}}$ over $O(1)$-radius balls in minor-free graph G, while not necessarily itself minor-free, must have *polynomial expansion* [50]. Without going into definitions, the class of graphs with polynomial expansion coincide with those that have sublinear-size separators for all subgraphs [21]. Thus one can apply the same kind of exchange argument using separator decomposition on the ball intersection graph $G_{\mathcal{R}}$.

Arbitrary radii and arbitrary weights. The r-dominating set problem becomes significantly harder when r can depend on n [22]. In the literature this is usually refer to as the the *metric/unbounded r-dominating set problem* [26, 27], where we further allow arbitrary nonnegative *edge-length ℓ* on the input graph, in addition to the covering radius r (now measuring with respect to ℓ) becoming part of the input. We can even consider a weighted variant of the problem, where every vertex is associated with a positive-valued *weight w*, and the goal is to compute an r-dominating set with minimum total weight (instead of minimum cardinality). Baker's layering technique no longer applies, nor does the local search strategy for the most part. (There is an exception [42, 26] which we will talk about in the technical section.) This is somewhat expected because the unbounded r-dominating set problem is no longer "local", and the intersection graph of radius-r balls may be dense and have complex interactions. In fact, under Exponential-Time Hypothesis, Fox-Epstein *et al.* [44, 28] proved that no efficient PTAS can exist for the unbounded r-dominating set problem on planar graphs. Thus, recent work has resorted to *bicriteria* algorithms, allowing the returned solution to use a dominating set of radius $(1+\varepsilon)r$ instead, in addition to the total weight being a $1+\varepsilon$ approximation to the optimal value [22, 28]. The idea is to embed the original planar/minor-free graph metric into some other *tree-like metrics*, where the r-dominating set problem can easily be solved using dynamic programming. Unfortunately one has to suffer some form of distortion (which translates to the slack in covering radius); the best result so far gives $+\varepsilon\Delta$-additive embedding of diameter-Δ minor-free graphs into graphs of $O_\varepsilon(1)$ treewidth [13, 14]. This line of research has been fruitful and ultimately leads to bicriteria EPTAS for planar graphs [28] and bicriteria Q(Q)PTAS for minor-free graphs [26, 27].

But what if we really want to solve the unbounded r-dominating set problem without compromise on the radius r? The lower bound result [44, 28] leaves us no choice but to look at $O(1)$-approximation (or at least, $(1+\varepsilon)$-approximation with running time $n^{f(\varepsilon)}$). The existence of such algorithm was explicitly asked by Filtser and Le [27].

Our contribution. We establish the first polynomial-time $O(1)$-approximation algorithm for the (vertex)-weighted metric r-dominating set problem for unbounded r. We emphasize that r is part of the input, and can be arbitrarily large compared to n.

▶ **Theorem 1.** *Let G be an arbitrary n-node planar graph, where each vertex v is associated with a weight $w(v)$. Let r be an arbitrary radius parameter. There is a polynomial-time algorithm that computes an r-dominating set in G whose weight is $O(1)$ times the optimum.*

1.1 Techniques

Our new (single-criteria) $O(1)$-approximation algorithm for weighted r-dominating set on planar graphs uses the quasi-uniformity sampling technique, originally designed for weighted set cover problem on geometric shapes [63, 11]. But first, we have to describe a bit of (technical) history and two immediate challenges when adapting to our setting.

Challenge 1: Planar support. Geometric set cover problems have their own history and success [7, 16, 17, 62]. Notably, for an unweighted set cover instance $(X, \mathcal{R})$, the result of Brönnimann and Goodrich [7] provided a recipe to achieve $O(1)$-approximation. Their pioneering use of multiplicative weight update algorithm requires the existence of small ε-*net*: First, find an ε-net N for the dual set system $(\mathcal{R}, X)$ (notice that here the ε-net is in the dual and thus a subset of ranges in $\mathcal{R}$). If there are element x in X not covered by N, *double* the weight of every range in $\mathcal{R}$ that covers x, to increase the chance of choosing one such range in the next round. As a result, by setting $\varepsilon = \Theta(1/\mathrm{opt})$, this implies an $O(g(\mathrm{opt}))$-approximation for any set cover instance, given the dual system has an ε-net of size $O(\frac{1}{\varepsilon} \cdot g(\frac{1}{\varepsilon}))$. (See [24] for an LP interpretation and a simpler proof.)

For set system of bounded VC-dimension [61], there is always an ε-net of size $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ [36]. Also, if the VC-dimension of a primal system is d, then the dual system has VC-dimension at most 2^d [34]. To get $O(1)$-approximation, however, we need a much stronger assumption of $O(\frac{1}{\varepsilon})$-size ε-net *in the dual system*. This is not always achievable even for simple geometric shapes like lines and rectangles [49]; in fact, among the basic geometric shapes, only a handful are known to have linear-size ε-net, including halfspaces in 2D and 3D [46, 51, 45], unit-cubes in 3D [17], and family of pseudodisks in the plane (both primal and dual) [53].

For our application of unbounded r-dominating set problem, the "geometric shapes" are the radius-r balls in planar graphs, which is discrete in nature, lacking well-defined boundary curves to even make the definition of pseudodisks meaningful. One might sneer at the issue and say, "there must be a way; how can distance balls in the plane be much different from pseudodisks?" But even if we redefine the notion of pseudodisks combinatorially (for every pair of regions A and B, all three sets $A \cap B$, $A \setminus B$, and $B \setminus A$ are connected), the distance balls are still not pseudodisk. In fact, they can be *piercing* in the sense of Raman-Ray [54], as illustrated by Figure 1.

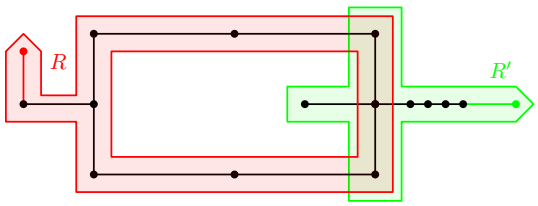


Figure 1 Two distance balls R and R' that are piercing. Here both R and R' have radius 6.

Fortunately, Pyrga and Ray [53] provided an alternative method to obtain linear-size ε-net (in the dual) – to construct a *sparse support* for the primal system $(X, \mathcal{R})$. We say a set system $(X, \mathcal{R})$ has a *support*, if for any subsystem $(X', \mathcal{R}_{|X'})$ of $(X, \mathcal{R})$, there is an undirected graph H on the elements of X' such that for every set R in $\mathcal{R}_{|X'}$, the subgraph H_R of H induced by the elements of R is connected. The support is *sparse* if every such graph H is sparse. The work of Pyrga and Ray [53] implies that if $(X, \mathcal{R})$ has bounded VC-dimension and a sparse support, then the dual system $(X, \mathcal{R})$ has ε-net of size $O(\frac{1}{\varepsilon})$.

Our first technical contribution is to compute a support for any collection of distance balls in any *arbitrary* graph (not just planar graphs). Crucially, if the given graph is planar, then so is the support. We remind the reader that our original goal is to solve the r-dominating set problem with unbounded r; thus the construction of the support has to hold for balls of unbounded radii as well. This is the first construction of its kind; most previous work on sparse support focus on geometric shapes, and almost entirely restricted to *planar support* for set systems that are pseudodisks [53] and their generalizations [54]. For piercing systems like axis-parallel rectangles, there are configurations where no planar support can exist [52].

Instead of focusing on planarity, we shift the narrative by arguing that a support graph for $(X, \mathcal{R})$ is really a *connectivity sketch* that exists for every graph G on X, as long as the sets in $\mathcal{R}$ are *distance balls* on G. Our construction for support on arbitrary balls in G can be described simply as follows: Build a *Voronoi diagram* of $V(G)$ with respect to balls centers, then contract every Voronoi cell into a single node. We argue that the contracted graph is a support of G. (The benefit of this additional property becomes evident when we tackle the next challenge.) Both planarity and sparseness come as a byproduct of the fact that contraction of the Voronoi diagram is a *minor* of G. (See Theorem 2 for a precise statement.) (A similar Voronoi diagram idea can be found in the work of Le [42, 26]; but their version only enables single-criteria algorithm for *unweighted* metric r-dominating set problem.)

Challenge 2: Linear shallow cell complexity. There is another, perhaps more severe, obstacle to overcome. We want to solve the *weighted* version of the r-dominating set problem. The Brönnimann-Goodrich MWU approach relies on computing regular ε-net (as subset of balls covering the vertices); the *size* bound on ε-net translates to approximation guarantee on *size* of the cover, and thus can only work in the unweighted setting. Varadarajan [63] extended the ε-net idea to the weighted setting. This *quasi-uniformity sampling technique* computes ε-nets that samples each set in $\mathcal{R}$ with (roughly) equal probability, which then can be made aware of the weights on $\mathcal{R}$. Chan, Grant, Könemann, and Sharpe [11] later polished this technique and made connection to a more refined measure on set systems called *shallow cell complexity*. In our context, cell complexity refers to the number of different equivalent classes (called *cells*) of points that are covered by the same subset of r-balls. We say a cell has *depth* k if the points within are covered by up to k balls. A set system has shallow cell complexity $f(|\mathcal{R}|, k)$ if the number of cells of depth at most k is bounded by $f(|\mathcal{R}|, k)$; for the purpose of obtaining $O(1)$-approximation for set cover, one requires the shallow cell complexity to be $O(|\mathcal{R}| \cdot \operatorname{poly} k)$.

The quasi-uniformity sampling technique can be used to solve the *constant* r case for weighted r-dominating set problem on minor-free graphs (and graphs with bounded expansion as well [20]). This is because in the case of constant radius r, the *whole* cell complexity is linearly bounded [57, 38], which directly bounds the shallow counterpart. Unfortunately, it is known that the cell complexity of r-balls on a planar graph can be cubic in r [38], and thus cannot help in the unbounded r setting.

One way to see why halfspaces and pseudodisks are special is that they both have linear *union complexity* [40, 17]. Application of the classical Clarkson-Shor technique then implies that the shallow cell complexity is bounded by $O(|\mathcal{R}| \cdot k)$. When working with arbitrary-radius balls on graphs however, we run into trouble applying Clarkson-Shor. First, there is no well-defined notion of union complexity for a collection of subgraphs. One would imagine the support can take the role as the linear-size object at the base case for Clarkson-Shor; but this intuition is insufficient. The main issue is that for pseudodisks, the number of cells can be charged to the complexity of the whole pseudodisk arrangement by Euler's formula, and every crossing in the arrangement is uniquely defined by two pseudodisks. There is no immediate equivalent object for distance balls that allows for such counting.

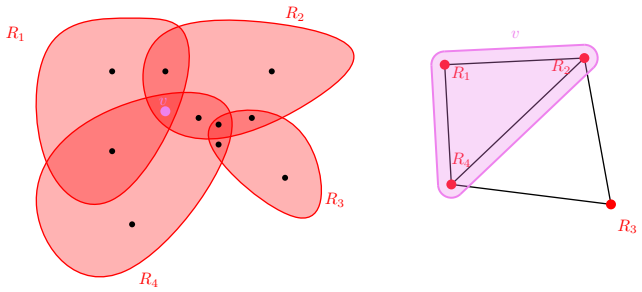


Figure 2 A set system with a dual support. The red regions containing the violet vertex v form a connected subgraph in the support.

Our solution is to reimagine the unique representative argument by Raman and Ray for the non-piercing family [55]; the new proof is substantially different and relies on geometric properties of distance balls, instead of topological properties of curve sweeping [55, §6]. First we argue that every cell can be charged to a unique 3-tuple of ball centers (Lemma 7). This allows us to apply Clarkson-Shor and reduce the problem to counting cells of depth 1, 2, and 3. Depth 1 and 2 are straightforward; for depth-3 cells, we exploit the additional property that our support for distance balls came from contracting *Voronoi cells* with respect to the ball centers, and thus the number of depth-3 cells can be counted towards the complexity of the Voronoi diagram, which is asymptotically the same as the planar support (Lemma 13).

2 Terminology and Support Graphs

Let's first introduce the terminology needed for concrete discussion. Our support construction actually works for balls in an arbitrary *directed* graph. We refer to the digraph that our ball system lives on as the *underlying graph* G. Note that all results apply to balls in undirected graphs. A *ball* D can be defined using a pair (c_D, r_D) where c_D is a vertex of G called the *center*, and a *radius* r_D which is a real number. Ball D is then defined to be the set of all vertices v whose shortest path from c_D has length at most r_D.

Given a system $(U, \mathcal{R})$, the *dual system* $(\mathcal{R}, U)$ has the elements of U treated as subsets of $\mathcal{R}$: if element e in U is a member of set R in $\mathcal{R}$ in the primal system, then R is a member of *set* e in the dual system. We use the term *dual support* of $(U, \mathcal{R})$ to refer to a support graph of the dual system $(\mathcal{R}, U)$. (We give an example of a dual support in Figure 2.) Although our r-dominating set application demands primal support, the set system of balls *of equal radius* is self-dual, Thus, we do not differentiate the support for the primal from the dual, but instead describe the support for the dual system since it is conceptually easier.

In Section 3 we construct a dual support for any system of balls, by contracting cells of the Voronoi diagram of the underlying graph (with respect to ball centers) into a minor.

▶ **Theorem 2** (Support). *Any system of balls $\mathcal{R}$ on an underlying digraph G has a dual support $H(\mathcal{R})$ such that if G does not contain a minor K with minimum vertex degree two, then neither does $H(\mathcal{R})$.*

▶ Remark. Unfortunately the min-degree requirement cannot be dropped in general. For example, if G is P_2-minor-free, we have a set of disjoint vertices. But two balls can centered on the same vertex, and thus connected in the dual support, which implies the existence of an edge and so the support is not P_2-minor-free. Still the restriction is not severe as most of the interesting minor-free graph classes satisfy the min-degree condition, including K_h-minors for $h \geq 3$, grid minors, and all forbidden minors for bounded-genus graphs.

Since planar graphs can be classified by their excluded minors, these results imply a planar dual support for the ball system $\mathcal{R}$ on G. We then proceed to show that $\mathcal{R}$ has linear shallow cell complexity in Section 4.

▶ **Theorem 3** (Shallow Cell Complexity). *Let G be a planar graph. Any system of balls $(V_G, \mathcal{R})$ has shallow cell complexity $O(|\mathcal{R}| \cdot k^2)$.*

2.1 Applications

In addition to the result on computing unbounded r-dominating set in planar graphs, the supports also have their own implications. An important property of our support construction is that we only ever contract edges that are within radius r of a ball center. This means that if G is an unweighted graph with *polynomial expansion* and we are considering constant-radius balls, the support graph will also have polynomial expansion, allowing us to use the sublinear separators from Har-Peled and Quanrud [35] to get the same PTAS results for unweighted problems as Raman and Ray [54] achieved for non-piercing regions. If r is not bounded by a constant, we can achieve the same when G is minor-free [65].

Another natural use of supports is for problems relating to geodesic disks in a polygonal domain (possibly with holes). It is known that planar metrics and polygonal domains are *equivalent*: any system of geodesic disks in a polygonal domain in $\mathbb{R}^2$ can be represented as a system of balls in a planar graph, and vice versa [6]. By the equivalence, our techniques naturally give planar supports for geodesic disks in polygonal domains.

2.2 Related Work on Support Graphs

Van Cleemput [60] and Voloshina and Feinberg [64] used the existence of a planar support graph to give meaningful notion of planarity in the hypergraph setting. As such, support graphs have seen interest in (hyper)graph drawing community [37, 39, 8]. In the realm of optimization, Pyrga and Ray [53] used support graphs to construct linear-sized ε-nets for several set systems with bounded VC dimension, including halfspaces in $\mathbb{R}^3$ and pseudodisks (more generally, r-admissable regions) in $\mathbb{R}^2$. Here the support graphs were required to be *sparse* (i.e. with linearly many edges) in order to bootstrap existing ε-net sizes down to $O(\frac{1}{\varepsilon})$. Later on support graphs that are *planar* were used by Mustafa and Ray [48] to show a PTAS for various optimization problem in the geometric setting. Instead of ε-net, they exploited the existence of *sublinear-size* separators in planar graphs to show that an $O_\varepsilon(1)$-swap local search algorithm gives a $(1+\varepsilon)$-approximation for minimum geometric hitting set. The same paradigm of analyzing local search with separators on planar support was extended to other problems, such as maximum independent set and generalized set cover (including dominating set and hitting set) in pseudodisks and non-piercing regions [12, 19, 32, 54], terrain guarding [31, 2], and covering the boundary of a simple orthogonal polygon with rectangles [5]. (A survey on the many use of support can be found in Raman and Singh [56].)

3 Existence of Support Graph

To begin constructing the dual support for a system of balls $\mathcal{R}$ in a graph G, we will first modify G slightly and reduce to the case where *every ball in $\mathcal{R}$ has the same radius*. We will refer to the new graph as G'. (Recall we work with the general case of digraphs.) We augment digraph G by adding a new node corresponding to each ball in $\mathcal{R}$. (For clarity, we refer to original vertices of G as *vertices* and new vertices corresponding to $\mathcal{R}$ as *nodes*.) For each ball R in $\mathcal{R}$, we create a single node, x_R, which has only one edge: an outgoing edge

into c_R, the center of the ball R. We give the edge a weight of $r_{\max} - r_R$, where $r_{\max}$ is defined to be $\max_{R' \in \mathcal{R}} r_{R'}$. This allows us to instead consider each R in $\mathcal{R}$ to be centered at x_R, all with the same radius $r_{\max}$, and the balls will still define the same set system $\mathcal{R}$ over the vertices in G. Finally, remove all vertices which are not contained in any ball of $\mathcal{R}$ and slightly perturb the edge weights (without changing any ball containment relations) so that every vertex v has a unique closest ball center in $\{x_R : R \in \mathcal{R}\}$.

To build the dual support, we construct a *Voronoi partition* of G' with respect to the set of nodes. A *Voronoi partition* $\mathcal{F}$ of a graph G' with respect to a set of nodes S is a partition of $V_{G'}$ into subsets such that for every node $R \in S$, there is a corresponding subset $F_R \in \mathcal{F}$ which contains all vertices u in G' for which $d(x_R, u) \leq \min_{x_R \in S \setminus \{x_R\}} d(x_{R'}, u)$. For clarity, we will refer to each set F_R in $\mathcal{F}$ as the *Voronoi cell* of R. In our case, we construct the Voronoi partition $\mathcal{F}_{\mathcal{R}}$ of G' with respect to $\{x_R : R \in \mathcal{R}\}$, which we identify as $\mathcal{R}$. Each cell F_R in the Voronoi partition is a connected subset of G' because for every vertex v in F_R reachable from x_R, the vertices on the shortest path between x_R and v must belong to F_R as well. Thus we can simply contract every F in $\mathcal{F}_{\mathcal{R}}$ to a (contraction)-minor on only the nodes in $\mathcal{R}$. This final graph H, we claim, is a dual support for $(G, \mathcal{R})$.

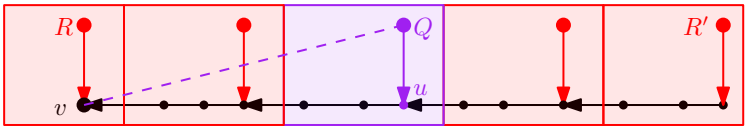


Figure 3 An illustration of the proof to Lemma 4 that π can only pass through Voronoi cells of $\mathcal{R}_v$. The purple region is the Voronoi cell F_Q.

▶ **Lemma 4.** *For any vertex v in G', the set of balls $\mathcal{R}_v \subseteq \mathcal{R}$ that contain v must induce a connected subgraph in H.*

Proof. Consider any vertex v which is contained in at least two balls in $\mathcal{R}$. Vertex v must be in the Voronoi cell F_R of some ball R by construction of G'. We claim that for any other ball R' of $\mathcal{R}$ that also contains v, there is a path from $x_{R'}$ to v in G', denoted as π, that only passes through Voronoi cells corresponding to balls that contain v.

Since we have removed all vertices which have no incoming path from any ball, every vertex along π in G' must be in some cell. Assume that some vertex u along π is contained in some Voronoi cell F_Q corresponding to a ball Q that does not contain v. By definition of Voronoi cell, we have that $d(x_Q, u) \leq d(x_{R'}, u)$. By the definition of shortest paths, we know that $d(x_{R'}, v) = d(x_{R'}, u) + d(u, v)$. By triangle inequality, we then have the following:

$$d(x_Q, v) \leq d(x_Q, u) + d(u, v) \leq d(x_{R'}, u) + d(u, v) = d(x_{R'}, v).$$

But, $d(x_{R'}, v) \leq r_{\max}$ and $d(x_Q, v) > r_{\max}$ since v is in R' but not in Q, contradiction. ◀

Proof of Theorem 2. Construct H as described above. By Lemma 4, we know that H satisfies the definition of a dual support for system $(V_G, \mathcal{R})$. It remains to show that if G has no minor K with minimum degree two, then neither can H. This is done by arguing that G' has no such minors, and then the fact that H is a minor of G' concludes the proof.

When constructing G', we only perform two types of operations that would affect the existence of minors: deleting vertices and adding in nodes $\{x_R\}$ of degree one. Let K be a minor model (as a collection of disjoint vertex subsets called *supernodes*) of G' that has minimum degree two. If K only contains vertices but not nodes in G', then K must appear in G as well (even though G' was obtained by removing some vertices from G). So at least one supernode η of K must contain some degree-1 x_R that was added in G'.

For an edge (η, η') to exist in K, there must be a vertex/node $u \in \eta$ and a vertex/node $u' \in \eta'$ such that (u, u') is an edge of G'. For η containing x_R, x_R must not be the only vertex/node in η, since η have degree at least 2. Because η is connected, the neighbor of x_R is also in η and the (sole) edge incident on x_R must be contracted. This, however, means that there can be no edge (x_R, u') in G' that corresponds to an edge (η, η') in K, so there is a minor K' isomorphic to K where x_R was simply deleted and thus x_R is not in η. Since this holds for all x_R, there must be some minor of G isomorphic to K, a contradiction. ◀

4 Shallow Cell Complexity of Balls in Planar Graphs

Shallow cell complexity was introduced by Chan *et al.* [11] as a combinatorial analog of union complexity. We introduce the terminology within our context of distance balls on graphs.

Define the *hit set* of a vertex v with respect to $\mathcal{R}$ to be the subset of balls in $\mathcal{R}$ which contain v. A *cell* of the system $(V_G, \mathcal{R})$ is a set of vertices in V_G that have the same hit set of incident balls. *Cell complexity* is simply the number of cells in $(V_G, \mathcal{R})$, where as *shallow* cell complexity counts the number of cells of a particular *depth* k in $\mathcal{R}$, denoted $Cell_k$, the equivalence classes of vertices whose hit set has size k. We say the set system has *shallow cell complexity* $f(n, k)$ if for depth k, there are at most $f(n, k)$ cells. Our goal for the section is to show that for a planar graph G, the shallow cell complexity for any system of balls $(V_G, \mathcal{R})$ is $O(|\mathcal{R}| \cdot k^2)$ (Theorem 3). Just as in the support constructions in § 3, we again reduce to the case where *every ball in our system* $\mathcal{R}$ *has the same radius.*

Proof of Theorem 3. We use two key lemmas proven later in the subsections. The first claims that for every cell c of depth k, there is a unique encoding of c, $\langle \alpha(c), \beta(c), \gamma(c) \rangle$, where each member of the tuple is a ball of $\mathcal{R}$ (Lemma 7 in Section 4.1). Importantly, the encoding is unique among cells of depth k, not universally. This naively implies that there is at most $O(|\mathcal{R}|^3)$ cells of depth k. Using the standard Clarkson–Shor arguments argument, we can improve this to $O(|\mathcal{R}| \cdot k^2)$ if the number of cells of depth (at most) 3 is $O(|\mathcal{R}|)$ (Lemma 13 in Section 4.2). More precisely: Sample the regions of $\mathcal{R}$ with probability $1/k$. A cell c with unique encoding appears in the sample as depth-3 cell with probability at least $(ek)^{-3}$. The expected number of depth-3 cells in the sample is at most $O(|\mathcal{R}|/k)$, so the number of depth-k cells in $\mathcal{R}$ is at most $O(|\mathcal{R}| \cdot k^2)$. ◀

4.1 Unique 3-Site Encoding of Cells

We begin by assigning each cell of $Cell_k$ an *unique* encoding for each fixed depth $k \geq 3$. This encoding will depend on the set of incident balls corresponding to c. Specifically, the encoding will be formed by a 3-tuple of regions in $\mathcal{R}$, which we will refer to as $\langle \alpha, \beta, \gamma \rangle$. In assigning the encoding, we do not modify to the set system at all. Instead, we fix an arbitrary vertex v_c for cell c as its *representative*. We denote the cell containing v as $Cell(v)$.

For a cell c, we begin by designating its $\langle \alpha, \beta \rangle$*-type*. We define $\alpha(v)$ as the furthest ball from v that contains v. Similarly, we define $\beta(v)$ as the second furthest ball from v that contains v. For cell c, its $\langle \alpha, \beta \rangle$*-type* is simply $\langle \alpha(v_c), \beta(v_c) \rangle$. (Notice that $\alpha(v_c)$ and $\beta(v_c)$ might be different if one chooses a different v_c from cell c, and thus the $\langle \alpha, \beta \rangle$-type of c depends on its representative.) Let $\pi_{\alpha,\beta}(v)$ be the union of $\pi(\alpha(v), v)$ and $\pi(\beta(v), v)$, where $\pi(x, y)$ is the shortest path from x to y in G.

▶ **Lemma 5.** *For a vertex v with $\alpha(v) = D_1$ and $\beta(v) = D_2$, every vertex v' in $\pi_{\alpha,\beta}(v)$ with $\alpha(v') = D_1$ and $\beta(v') = D_2$ satisfies $hit(v') \subseteq hit(v)$.*

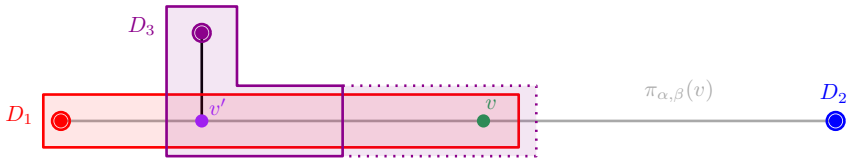


Figure 4 Given two vertices v, v' satisfying $\alpha(v) = \alpha(v') = D_1$, $\beta(v) = \beta(v') = D_2$ and $v' \in \pi(D_1, v)$, any ball D_3 containing v' must also contain v.

Proof. Assume the contrary. Then, there is some vertex v' which is contained within some ball D_3 such that $v \notin D_3$. By definition of α and β, it must be that the center of D_3 is closer to v' than D_1 and D_2. However, since v' is either in $\pi(c_{D_1}, v)$ or $\pi(c_{D_2}, v)$, this means that D_3 is closer to v than one of D_1 or D_2, and thus D_3 must contain v, a contradiction by the assumption that all balls have equal radius. (See Figure 4.) ◀

The third component of our encoding, γ, can now be defined. For $k = 3$, γ is simply the remaining ball after removing α and β. For $k \geq 4$, we define γ procedurally after fixing α, β. Denote the set of all cells of depth k with $\alpha(v_c) = D_1$ and $\beta(v_c) = D_2$ as $C_k(D_1, D_2)$. We additionally collect the representatives v_c of each $c \in C_k(D_1, D_2)$ into a corresponding set $V_k(D_1, D_2)$. Let $T_D(S)$ be the shortest path tree from the center x_D of a ball D to a set of vertices S. Consider the shortest path tree $T_{D_1}(V_k(D_1, D_2))$. We will process all $v_c \in V_k(D_1, D_2)$ using the plane drawing of $T_{D_1}(V_k(D_1, D_2))$.

Perform a depth-first traversal starting at x_D, processing children in clockwise order. The depth-first traversal $T_{D_1}(V_k(D_1, D_2))$ of gives a total ordering of the vertices in $V_k(D_1, D_2)$, which also induces a total ordering σ on the corresponding cells $C_k(D_1, D_2)$. For brevity, we will index the cells based on their order in σ (the ith cell as c_i) in a cyclic manner. For the ith cell c_i in the total ordering, we assign $\gamma(c_i)$ to be an arbitrary ball D with $D \in hit(v_{c_i})$ and $D \notin hit(v_{c_{i-1}})$. Since $|hit(v_{c_i})| = |hit(v_{c_i-1})| = k$ and $hit(v_{c_i}) \neq hit(v_{c_i-1})$ (by the definition of cell), such a D must always exist, and thus we can assign $\gamma(c_i)$ to all cells.

We now show that this definition of γ is sufficient to ensure that each distinct cell c of size k has a unique encoding. First, we use Lemma 5 to show that the representative vertices of cells with the same α and β cannot form ancestor relationships in $T_{D_1}(V_k(D_1, D_2))$.

▶ **Observation 6.** *For any two distinct cells $c, c' \in C_k(D_1, D_2)$, v_c cannot be an ancestor of $v_{c'}$ in $T_{D_1}(V_k(D_1, D_2)$ or $T_{D_2}(V_k(D_1, D_2)$.*

Proof. First consider $T_{D_1}(V_k(D_1, D_2))$. If v_c were an ancestor of $v_{c'}$, then v_c would be in $\pi(D_1, v_{c'})$. This would imply that $hit(v_{c'}) \subseteq hit(v_c)$ by Lemma 5. However, since c and c' have depth k, $|hit(v_c)| = |hit(v_{c'})| = k$. The subset relationship would hence require that $hit(v_c) = hit(v_{c'})$, and thus c and c' are the same class, which is a contradiction. A symmetric argument applies for $T_{D_2}(V_k(D_1, D_2))$. ◀

This means that for any c, c' with the same α and β, $\pi(\alpha(v_c), v_c) \not\subseteq \pi(\alpha(v_c), v_{c'})$.

▶ **Lemma 7.** *For $k \geq 3$, every cell c of depth k has a unique encoding $\langle \alpha(c), \beta(c), \gamma(c) \rangle \in \mathcal{R}^3$.*

Proof. If two cells have the same depth and α and β, they are both indexed by the same traversal order σ. Consider two cells c_i, c_j such that $\langle \alpha(c_i), \beta(c_i), \gamma(c_i) \rangle = \langle \alpha(c_j), \beta(c_j), \gamma(c_j) \rangle = (D_1, D_2, D_3)$ and $|hit(v_{c_i})| = |hit(v_{c_j})| = k$.

We need to consider how c_i and c_j were assigned the same γ. Recall that we force adjacent cells c_{i-1}, c_i to not just have different γ assignments, but that the $\gamma(c_i)$ not contain $\gamma(c_{i-1})$. This forces $v_{c_{i-1}}$ and $v_{c_{j-1}}$ to not be contained by D_3. Let a be the lowest common ancestor

of $v_{c_{i-1}}$ and $v_{c_{j-1}}$ in $T_{D_1}(V_k(D_1, D_2))$. Similarly, let b be the lowest common ancestor of $v_{c_{i-1}}$ and $v_{c_{j-1}}$ in $T_{D_2}(V_k(D_1, D_2))$. We consider the cycle F formed by the union of the following shortest paths: $\pi(a, v_{c_{i-1}})$ to $\pi(v_{c_{i-1}}, b)$ to $\pi(b, v_{c_{j-1}})$ to $\pi(v_{c_{j-1}}, a)$. Since G is planar, F describes the boundary of some region $R(F)$. Since $i-1 < i < j-1 < j$, by the definition of our traversal one of v_{c_i} or v_{c_j} lies in $R(F)$ and the other lies outside. (We give an example $R(F)$ as a gray region in Figure 5.) This means that, for D_3 to contain both v_{c_i} and v_{c_j}, D_3 must also contain some vertex u' of F. Assume WLOG that u' lies on $\pi(D_3, v_{c_j})$. Since u' lies on F, it must be on one of $\pi(D_1, v_{c_{i-1}}), \pi(D_1, v_{c_{j-1}}), \pi(D_2, v_{c_{i-1}})$, or $\pi(D_2, v_{c_{i-1}})$.

- If u' is in $\pi(D_1, v_{c_{i-1}})$ or $\pi(D_1, v_{c_{j-1}})$, then we know that D_3 must be further from u' than D_1 since D_3 cannot contain $v_{c_{i-1}}$ or $v_{c_{j-1}}$. However, since $\alpha(v_{c_j}) = D_1$, D_3 must be closer to u' than D_1 since u' is on $\pi(D_3, v_{c_j})$ and $\alpha(v_{c_j})$ is the furthest ball from v_{c_j} which still contains it. (See left subfigure of Figure 5.)
- Similarly, if u' is in $\pi(D_2, v_{c_{i-1}})$ or $\pi(D_2, v_{c_{j-1}})$, then D_3 must be further from u' than D_2 since D_3 cannot contain $v_{c_{i-1}}$ or $v_{c_{j-1}}$. However, since $\beta(v_{c_j}) = D_2$, D_3 must be closer to u' than D_2 since u' is on $\pi(D_3, v_{c_j})$ and $\beta(v_{c_j})$ is the second furthest ball from v_{c_j}, while D_3 is at best third furthest. (See right subfigure of Figure 5.)

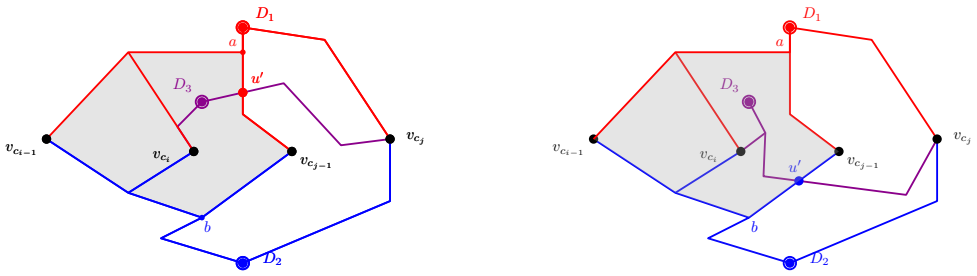


Figure 5 If two cells c_i, c_j have the same $\gamma(c_i) = \gamma(c_j) = D_3$, then there must D_3 must contain a vertex u' of $\gamma_{\alpha,\beta}(v_{c_{j-1}})$ or $\gamma_{\alpha,\beta}(v_{c_{j-1}})$. The region $R(F)$ corresponding to the simple cycle F is given in gray.
Left: u' is in $\pi(D_1, v_{c_{j-1}})$. Right: u' is in $\pi(D_2, v_{c_{j-1}})$.

We get a contradiction in both cases, which means that $\gamma(c_i)$ cannot equal $\gamma(c_j)$, and thus every cell c of the same depth $k \geq 3$ must have a unique encoding $\langle \alpha(c), \beta(c), \gamma(c) \rangle$. ◀

4.2 Depth-3 Cells

Now that we have a unique encoding $\langle \alpha(v_c), \beta(v_c), \gamma(v_c) \rangle$ for every cell c of depth k, we can proceed to give a bound on $|Cell_3|$.

For each c in $Cell_3$, we consider its encoding $\langle \alpha(v_c), \beta(v_c), \gamma(v_c) \rangle$. Since c only has depth 3, we know that these are simply the balls of $hit(v_c)$, placed in order of distance from v_c. Our first step is to select one of the three balls of $hit(v_c)$ as $base(v_c)$. Next, we iterate through every ball D in $\mathcal{R}$ and count the number of cells that have been assigned D. We accomplish this by constructing a planar graph G_D on the neighbors of D in the Voronoi partition graph H. Summing over all D, this yields a total count of $O(|\mathcal{R}|)$ cells. We start with an important lemma regarding the order of distances to balls from vertices that lie on a shortest path.

▶ **Lemma 8.** *Given any ball D of $\mathcal{R}$, a vertex v, some vertex v' on $\pi(x_D, v)$, and a distinct ball $D' \in \mathcal{R}$, if $d(x_{D'}, v') \leq d(x_D, v')$, then for all $v'' \in \pi(v', v)$, $d(x_{D'}, v'') \leq d(x_D, v'')$.*

Proof. For any $v'' \in \pi(v', v)$, because v' is between x_D and v'' on $\pi(x_D, v)$, $d(x_D, v'') = d(x_D, v') + d(v', v'')$. We also know that $d(x_{D'}, v'') \leq d(x_{D'}, v') + d(v', v'')$. Since $d(x_{D'}, v') \leq d(x_D, v')$, we can simply plug in to get $d(x_{D'}, v'') \leq d(x_D, v') + d(v', v'') = d(x_D, v'')$. ◀

▶ **Observation 9.** *For a cell c and ball $D' \in hit(v_c)$, all vertices v in $\pi(x_D, v_c)$ must lie in the Voronoi cell of a ball in $hit(v_c)$.*

Proof. Consider a ball $D' \notin hit(v_c)$ for which a vertex v of $\pi(x_D, v_c)$ lies in the Voronoi cell F_D. If this is the case, then all the vertices after v in $\pi(x_D, v_c)$ will be closer to D' than they are to D, and since D contains v_c, D' must also then contain v_c, which is a contradiction. ◀

This means that the shortest paths from the centers of the balls hit by v_c to the vertex v_c must stay within the Voronoi cells of $hit(v_c)$. This observation, paired with Lemma 8, allows us to define $base(v_c)$ based on $\pi(x_{\alpha(v_c)}, v_c)$. We consider two cases:

- *Case 1: Some vertex v in $\pi(x_{\alpha(v_c)}, v_c)$ lies in $F_{\beta(v_c)}$.* Assign $base(v_c) \leftarrow \beta(v_c)$.
- *Case 2: No vertex v in $\pi(x_{\alpha(v_c)}, v_c)$ lies in $F_{\beta(v_c)}$.* Assign $base(v_c) \leftarrow \gamma(v_c)$.

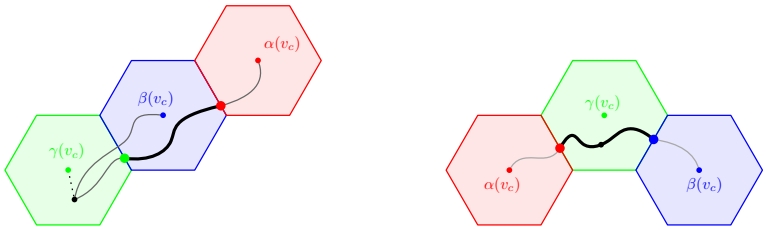


Figure 6 Showing the subwalk of Lemma 10 and how to select *base*. Left: Case 1. Right: Case 2.

Now that we have assigned every cell to a ball, we consider a ball D and the set $resident(D)$, which is the set of cells c with $base(c) = D$. We will consider the cells of $resident(D)$ in relation to the neighbors of D within the Voronoi partition graph H.

▶ **Lemma 10.** *For a ball D and every cell c of $resident(D)$, there is a subwalk π_c of $\pi_{\alpha,\beta}(v_c)$ with the (nonempty set of) interior vertices lying in D and the end vertices lying in distinct neighbors D', D'' of D in H.*

Proof. If cell c is a *case-1* cell, we have that a vertex v in $\pi(x_{\alpha(v_c)}, v_c)$ which lies in $F_{\beta(v_c)}$. By Lemma 8, no vertex of $\pi(v, v_c)$ can lie in $F_{\alpha(v_c)}$. Additionally, v_c must lie in $F_{\gamma(v_c)}$ by definition of α and β. Let u be the first vertex of $\pi(v, v_c)$ which lies in $F_{\gamma(v_c)}$. Let v' be last vertex of $\pi_{\alpha,\beta}(v_c)$ which lies in $F_{\alpha(v_c)}$. The subwalk $\pi_{\alpha,\beta}(v_c)[v', u]$ satisfies this condition.

If cell c is a *case-2* cell, we have no such vertex and $\pi(x_{\alpha(v_c)}, v_c)$ lies entirely in $\alpha(v_c)$ or $\gamma(v_c)$. Let u be the last vertex of $\pi(x_{\alpha(v_c)}, v_c)$ which lies in $F_{\alpha(v_c)}$ and let u' be the last vertex of $\pi(x_{\beta(v_c)}, v_c)$ that lies in $F_{\beta(v_c)}$. We then can take our subpath to be $\pi_{\alpha,\beta}(v_c)[u, u']$, and this will satisfy the condition. ◀

We now consider the neighbors of D in the Voronoi support H, $N_H(D)$. In Lemma 10, the subwalk π_c of cell c in $resident(D)$ runs between the two balls of $hit(v_c) \setminus D$, both of which must be in $N_H(D)$ since H is simply the contraction of the Voronoi partition of G. We can now define a graph G_D on $N_H(D)$ to count the number of cells in $resident(D)$. For every cell c of $resident(D)$, add an edge e_c between the balls of $hit(v_c) \setminus D$. Since the underlying graph G is planar, we can sketch a drawing of G_D as follows:

To draw the curve ζ_c corresponding to cell c of $resident(D)$, we first consider the subpath π_c that we get from Lemma 10, using a plane drawing of G_D. Let v be the first endpoint of π_c lying in a ball D_1 and v' be the second endpoint of π_c lying in a ball D_2. We first draw

the center x_{D_1} in its location on any plane drawing of the underlying graph G. We then sketch ζ_c on $\pi(x_{D_1,v})$. This is followed by sketching π_c, and then finally sketching the walk $\pi(v', x_{D_2})$. Since this sketch is over a path, we will refer to this walk in G as $\pi(\zeta_c)$.

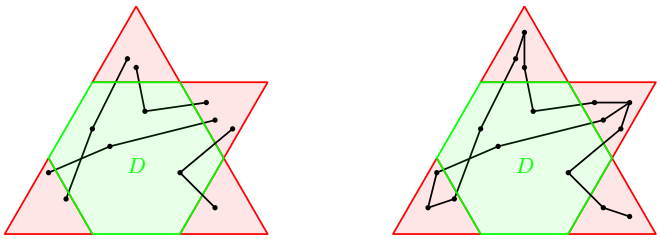


Figure 7 Left: a set of subwalks of cells in $resident(D)$. Right: the walks $\pi(\zeta_c)$ for each $c \in resident(D)$.

▶ **Lemma 11.** *For a given ball D in $\mathcal{R}$ and two distinct cells c and c' in $resident(D)$, if $\pi(\zeta_c)$ and $\pi(\zeta_{c'})$ share a common vertex v, then e_c and $e_{c'}$ share a common endpoint D'.*

Proof. Let $e_c = (D_1, D_2)$ and $e_{c'} = (D_3, D_4)$. For c (c'), the common vertex v must lie on a shortest path from either x_{D_1} or x_{D_2} (x_{D_3} or x_{D_4}) to v_c ($v_{c'}$) by construction. Without loss of generality, assume x_{D_1} and x_{D_3}. One of x_{D_1} or x_{D_3} must be at least as close as the other to v_c. Without loss of generality, assume x_{D_1}. By Lemma 8, this implies that D_1 contains $v_{c'}$, and is thus in $hit(v_{c'})$. We know that D is contained within $hit(v_{c'})$, as we only ever assign $base(v_{c'})$ to a member of $hit(v_{c'})$. Thus, in order for c' to be depth 3, D_1 must be either the same as D_3 or D_4. Thus, e_c and $e_{c'}$ have a common endpoint. ◀

Lemma 11 implies that ζ_c and $\zeta_{c'}$ only cross if e_c and $e_{c'}$ share an endpoint. Since the only crossings occur between edges incident on the same vertex, G_D can be redrawn with these crossings removed. This immediately implies the following:

▶ **Lemma 12.** *G_D is a simple planar graph.*

▶ **Lemma 13.** *There are $O(|\mathcal{R}|)$ cells of depth at most 3.*

Proof. We first quickly argue that there is $O(n)$ cells of depth at most 2. By the definition of cell, there is at most n cells of depth 1, one for each ball in $\mathcal{R}$. By the existence of a planar dual support (Theorem 2) on $\mathcal{R}$ with respect to all vertices of G, we know that for every cell of depth 2 corresponding to vertices v incident on balls R and R', the connectivity of $\mathcal{R}_v$ requires that R and R' share an edge. Since the dual support is planar, there is at most $3|\mathcal{R}| - 6$ such pairs, and thus at most $O(n)$ cells of depth 2.

For depth-3 cells, every edge of G_D corresponds to a resident depth-3 cell c of D, so $|resident(D)| \leq |E(G_D)|$. By Lemma 12, $|resident(D)| \leq 3|V(G_D)|$. By charging the depth-3 cells to the total number of edges across all G_D for all $D \in \mathcal{R}$ using the planarity of G_D, $|resident(D)| \leq 3|N_H(D)|$. Summing over all neighborhoods in the contracted Voronoi diagram of $\mathcal{R}$ in G, we know the sum of degrees is at most $6|\mathcal{R}|$. Thus, $|Cell_3| \leq 18|\mathcal{R}|$. ◀

References

1 Brenda S. Baker. Approximation algorithms for np-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, January 1994. doi:10.1145/174644.174650.

2 Sayan Bandyapadhyay and Aniket Basu Roy. Effectiveness of local search for art gallery problems. In Faith Ellen, Antonina Kolokolova, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures*, pages 49–60, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-62127-2_5.

3 J. Barilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *Journal of Algorithms*, 15(3):385–415, November 1993. doi:10.1006/jagm.1993.1047.

4 Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 184–193, Burlington, VT, USA, 1996. IEEE Comput. Soc. Press. doi:10.1109/SFCS.1996.548477.

5 Aniket Basu Roy. Covering simple orthogonal polygons with rectangles. In Alina Ene and Eshan Chattopadhyay, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2025)*, volume 353 of *Leibniz International Proceedings in Informatics (Lipics)*, pages 2:1–2:23, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2025.2.

6 Sujoy Bhore, Balázs Keszegh, Andrey Kupavskii, Hung Le, Alexandre Louvet, Dömötör Pálvölgyi, and Csaba D. Tóth. Spanners in planar domains via steiner spanners and non-steiner tree covers. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4292–4326. SIAM, 2025. doi:10.1137/1.9781611978322.145.

7 H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, December 1995. doi:10.1007/BF02570718.

8 Kevin Buchin, Marc Van Kreveld, Henk Meijer, Bettina Speckmann, and Kevin Verbeek. On planar supports for hypergraphs. In *International Symposium on Graph Drawing*, pages 345–356. Springer, 2009. doi:10.1007/978-3-642-11805-0_33.

9 Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. Sparse covers for planar graphs and graphs that exclude a fixed minor. *Algorithmica*, 69(3):658–684, July 2014. doi:10.1007/s00453-013-9757-4.

10 Sergio Cabello and David Gajser. Simple ptas's for families of graphs excluding a minor, March 2015. doi:10.48550/arXiv.1410.5778.

11 Timothy M. Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1576–1585. Society for Industrial and Applied Mathematics, January 2012. doi:10.1137/1.9781611973099.125.

12 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, September 2012. doi:10.1007/s00454-012-9417-5.

13 Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenković, Shay Solomon, and Cuong Than. Covering planar metrics (and beyond): O(1) trees suffice, June 2023. doi:10.48550/arXiv.2306.06215.

14 Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenković, Shay Solomon, and Cuong Than. Shortcut partitions in minor-free graphs: Steiner point removal, distance oracles, tree covers, and more. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5300–5331. SIAM, 2024. doi:10.1137/1.9781611977912.191.

15 M. Chlebík and J. Chlebíková. Approximation hardness of dominating set problems in bounded degree graphs. *Information and Computation*, 206(11):1264–1275, November 2008. doi:10.1016/j.ic.2008.07.003.

16 Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, ii. *Discrete & Computational Geometry*, 4:387–421, 1989. doi:10.1007/BF02187740.

17 Kenneth L. Clarkson and Kasturi Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007. doi:10.1007/s00454-006-1273-8.

18 Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k,r)-center in planar graphs and map graphs. *ACM Transactions on Algorithms*, 1(1):33–47, July 2005. doi:10.1145/1077464.1077468.

19 Stephane Durocher and Robert Fraser. Duality for geometric set cover and geometric hitting set problems on pseudodisks. In *CCCG*, 2015.

20 Zdeněk Dvořák. Approximation metatheorems for classes with bounded expansion. *LIPIcs, Volume 227, SWAT 2022*, 227:22:1–22:17, 2022. doi:10.4230/LIPIcs.SWAT.2022.22.

21 Zdenek Dvorak and Sergey Norin. Strongly sublinear separators and polynomial expansion, April 2015. arXiv:1504.04821.

22 David Eisenstat, Philip N. Klein, and Claire Mathieu. Approximating k-center in planar graphs. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 617–627. Society for Industrial and Applied Mathematics, January 2014. doi:10.1137/1.9781611973402.47.

23 David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, June 2000. doi:10.1007/s004530010020.

24 Guy Even, Dror Rawitz, and Shimon (Moni) Shahar. Hitting sets when the vc-dimension is small. *Information Processing Letters*, 95(2):358–362, July 2005. doi:10.1016/j.ipl.2005.03.010.

25 Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, July 1998. doi:10.1145/285055.285059.

26 Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. *arXiv:2101.01146 [cs]*, April 2021. arXiv:2101.01146.

27 Arnold Filtser and Hung Le. Low treewidth embeddings of planar and minor-free metrics. *arXiv:2203.15627 [cs]*, March 2022. doi:10.48550/arXiv.2203.15627.

28 Eli Fox-Epstein, Philip N. Klein, and Aaron Schild. Embedding planar graphs into low-treewidth graphs with applications to efficient approximation schemes for metric problems. In *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1069–1088. SIAM, January 2019. doi:10.1137/1.9781611975482.66.

29 Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, December 1987. doi:10.1137/0216064.

30 Cyril Gavoille, David Peleg, André Raspaud, and Eric Sopena. Small k-dominating sets in planar graphs with applications. In Gerhard Goos, Juris Hartmanis, Jan Van Leeuwen, Andreas Brandstädt, and Van Bang Le, editors, *Graph-Theoretic Concepts in Computer Science*, volume 2204, pages 201–216. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. doi:10.1007/3-540-45477-2_19.

31 Matt Gibson, Gaurav Kanade, Erik Krohn, and Kasturi Varadarajan. Guarding terrains via local search. *Journal of Computational Geometry*, 5(1):168–178, May 2014. doi:10.20382/jocg.v5i1a9.

32 Sathish Govindarajan, Rajiv Raman, Saurabh Ray, and Aniket Basu Roy. Packing and covering with non-piercing regions. In Piotr Sankowski and Christos Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (Lipics)*, pages 47:1–47:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2016.47.

33 Martin Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, 23(4):613–632, December 2003. doi:10.1007/s00493-003-0037-9.

34 Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, Rhode Island, June 2011. doi:10.1090/surv/173.

35 Sariel Har-Peled and Kent Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. *SIAM Journal on Computing*, 46(6):1712–1744, January 2017. doi:10.1137/16M1079336.

36 David Haussler and Emo Welzl. Epsion-nets and simplex range queries. *Discrete & Computational Geometry*, 2(2):127–151, June 1987. doi:10.1007/BF02187876.

37 D. S. Johnson and H. O. Pollak. Hypergraph planarity and the complexity of drawing venn diagrams. *Journal of Graph Theory*, 11(3):309–325, 1987. doi:10.1002/jgt.3190110306.

38 Gwenaël Joret and Clément Rambaud. Neighborhood complexity of planar graphs, February 2023. doi:10.48550/arXiv.2302.12633.

39 Michael Kaufmann, Marc van Kreveld, and Bettina Speckmann. Subdivision drawings of hypergraphs. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Graph Drawing*, pages 396–407, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

40 Klara Kedem, Ron Livne, János Pach, and Micha Sharir. On the union of jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete & Computational Geometry*, 1(1):59–71, March 1986. doi:10.1007/BF02187683.

41 Philip Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing - STOC '93*, pages 682–690, San Diego, California, United States, 1993. ACM Press. doi:10.1145/167088.167261.

42 Hung V. Le. *Structural Results and Approximation Algorithms in Minor-free Graphs*. PhD thesis, Oregon State University, 2018.

43 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, April 1979. doi:10.1137/0136016.

44 Dániel Marx and Michał Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015*, pages 865–877, Berlin, Heidelberg, 2015. Springer. doi:10.1007/978-3-662-48350-3_72.

45 Jiří Matoušek. Reporting points in halfspaces. *Computational Geometry: Theory and Applications*, 2(3):169–186, November 1992. doi:10.1016/0925-7721(92)90006-E.

46 Jiří Matoušek, Raimund Seidel, and E. Welzl. How to net a lot with little: Small ε-nets for disks and halfspaces. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, SCG '90, pages 16–22, New York, NY, USA, May 1990. Association for Computing Machinery. doi:10.1145/98524.98530.

47 Gary L Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32:265–279, 1986. doi:10.1016/0022-0000(86)90030-9.

48 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, December 2010. doi:10.1007/s00454-010-9285-9.

49 Nabil H. Mustafa and Kasturi R. Varadarajan. Epsilon-approximations and epsilon-nets, August 2017. doi:10.48550/arXiv.1702.03676.

50 Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion i. decompositions. *European Journal of Combinatorics*, 29(3):760–776, April 2008. doi:10.1016/j.ejc.2006.07.013.

51 János Pach and Gerhard Woeginger. Some new bounds for epsilon-nets. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, SCG '90, pages 10–15, New York, NY, USA, May 1990. Association for Computing Machinery. doi:10.1145/98524.98529.

52 Ambar Pal, Rajiv Raman, Saurabh Ray, and Karamjeet Singh. A fast algorithm for computing a planar support for non-piercing rectangles, October 2024. doi:10.48550/arXiv.2410.02449.

53 Evangelia Pyrga and Saurabh Ray. New existence proofs epsilon-nets. In *Proceedings of the 24th ACM Symposium on Computational Geometry, College Park, MD, USA, June 9-11, 2008*, Scg '08, pages 199–207, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1377676.1377708.

54 Rajiv Raman and Saurabh Ray. Constructing planar support for non-piercing regions. *Discrete & Computational Geometry*, 64(3):1098–1122, 2020. doi:10.1007/s00454-020-00216-w.

55 Rajiv Raman and Saurabh Ray. On the geometric set multicover problem. *Discrete & Computational Geometry*, 68(2):566–591, September 2022. doi:10.1007/s00454-022-00402-y.

56 Rajiv Raman and Karamjeet Singh. On hypergraph supports. In *European Conference on Combinatorics, Graph Theory and Applications*, pages 774–783, 2023.

57 Felix Reidl, Fernando Sánchez Villaamil, and Konstantinos Stavropoulos. Characterising bounded expansion by neighbourhood complexity. *European Journal of Combinatorics*, 75:152–168, January 2019. doi:10.1016/j.ejc.2018.08.001.

58 Neil Robertson and P.D Seymour. Graph minors. xvi. excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 89(1):43–76, September 2003. doi:10.1016/S0095-8956(03)00042-X.

59 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, November 2004. doi:10.1145/1039488.1039493.

60 W. M. van Cleemput. Hypergraph models for the circuit layout problem. *Applied Mathematical Modelling*, 1(3):160–161, 1976.

61 Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities, 1971.

62 Kasturi Varadarajan. Epsilon nets and union complexity. In *Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry*, pages 11–16, Aarhus Denmark, June 2009. ACM. doi:10.1145/1542362.1542366.

63 Kasturi Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, pages 641–648, Cambridge Massachusetts USA, June 2010. ACM. doi:10.1145/1806689.1806777.

64 A. A. Voloshina and V. Z. Feinberg. Planarity of hypergraphs. In *Doklady akademii nauk belarusi*, volume 28, pages 309–311, Minsk, Belarus, 1984. Academii Nauk Belarusi.

65 Christian Wulff-Nilsen. Separator theorems for minor-free and shallow minor-free graphs with applications. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 37–46, 2011. doi:10.1109/FOCS.2011.15.

The Typical Algebraic Shifting of Graphs and Surfaces

Denys Bulavka
Department of Mathematics & Statistics, Dalhousie University, Halifax, Canada

Eran Nevo
Institute of Mathematics, Universidad de Valladolid, Spain
Einstein Institute of Mathematics, Hebrew University, Jerusalem, Israel

Yuval Peled
Einstein Institute of Mathematics, Hebrew University, Jerusalem, Israel

Abstract

We initiate a statistical study of Kalai's exterior algebraic shifting, focusing on concentration phenomena for random triangulations of a fixed space. First, for a uniform n-vertex refinement of any given graph G, we show that asymptotically almost-surely (a.a.s.) its exterior algebraic shifting is an explicit shifted graph depending only on n and the Betti numbers of G. Next, for any given compact connected Riemannian surface S, sample n points independently at random according to the volume measure, and consider the resulted a.a.s. unique Delaunay triangulation. We prove that a.a.s. its exterior algebraic shifting is an explicit shifted complex depending only on n and the Euler genus of S, and in particular is area-rigid. In both results the expected shifted complex is a *homology lex-segment* complex, a notion we define combinatorially and characterize numerically à la Björner–Kalai.

As a tool to prove the result on surfaces, we prove a universality result on edge contractions: for every fixed surface triangulation K, every dense enough point set in the surface yields a Delaunay triangulation that edge contracts to K.

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces; Mathematics of computing → Random graphs

Keywords and phrases Algebraic shifting, Delaunay triangulation, surfaces, random triangulation, area rigidity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.25

Related Version *Full Version*: `https://arxiv.org/pdf/2509.26525` [5]

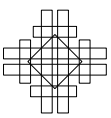
Funding *Denys Bulavka*: Partially supported by AARMS postdoctoral fellowship and by grant ISF-687/24.
Eran Nevo: Partially supported by the Israel Science Foundation grants ISF-2480/20 and ISF-687/24.
Yuval Peled: Partially supported by the Israel Science Foundation grant ISF-3464/24.

Acknowledgements We would like to thank the anonymous reviewers for their useful comments.

1 Introduction

A simplicial complex K on the vertex set $[n]$ is *shifted* if for all $1 \leq i < j$, $j \in A \in K$ implies that $(A \setminus \{j\}) \cup \{i\} \in K$. Erdős, Ko and Rado [11] introduced combinatorial shifting operators, which reduce problems on simplicial complexes to the shifted case. These operators have been used in many problems in extremal combinatorics, see Frankl's survey [13]. Kalai [16] introduced the exterior algebraic shifting operator on simplicial complexes, which is a canonical way to associate a shifted complex with a given simplicial complex, based on the exterior face ring over a fixed infinite field. Algebraic shifting has found many applications in combinatorics, especially in f-vector theory, and is interesting on its own, see, e.g., Kalai's survey [18].


© Denys Bulavka, Eran Nevo, and Yuval Peled;
licensed under Creative Commons License CC-BY 4.0



While algebraic shifting preserves the face numbers and Betti numbers of simplicial complexes, it is not determined by them in general, not even in dimension 1, see Example 1.5 below. However, it is determined for all n vertex triangulations of a fixed 1-dimensional compact manifold, by the properties mentioned above together with [24, Theorem 4.6] in order to combine the exterior algebraic shifting of each connected component. In dimension 2, for any fixed compact connected surface without boundary, all its n vertex triangulations have the same face numbers (and of course also the same Betti numbers). While the exterior algebraic shifting is constant on n vertex triangulations of the 2-sphere, see [18, Sec.2.4] or [19, Footnote 2], it is not constant for higher Euler genus surfaces; see [19] for a characterization of the possible shifted complexes for small Euler genus surfaces. Although the exterior algebraic shifting is not constant in these cases, it was observed in [19, Remarks 5.3] that many of the triangulations of the torus have the same exterior algebraic shifting. Naturally the following question arises: is there a *typical* exterior algebraic shifting of a surface triangulation?

We study the expected asymptotic behavior of the exterior algebraic shifting over triangulations of a fixed compact space X. Two natural random models come to mind: (1) fix the topology of the space and sample a random n-vertex triangulation $U_n(X)$ of it uniformly, or, (2) fix a metric and a volume form on the space and consider the random Delaunay model, where an n points set P_n is sampled at random according to the volume measure, and the unique Delaunay complex $\mathrm{Del}(P_n)$ they define is considered – this complex turns out to be an embedded triangulation of X, for X a Riemannian surface and n large enough [22, 10].

We analyze the first model for 1-dimensional spaces and the second model for closed connected Riemannian surfaces. (We will consider only surfaces without boundary.) In both cases we show *concentration*, namely, that a.a.s. (i.e., with probability tending to 1 as $n \to \infty$) the resulted shifted complex is the unique *homology lex-segment* on n vertices with the same homology as X, denoted by $\Delta(X, n)$. We describe $\Delta(X, n)$ for graphs and surfaces.

Let $G = (V, E)$ be a fixed graph, and $|G|$ denote its geometric realization, namely its corresponding topological space. Denote: c is the number of connected components of G, $|V| = m + c - 1$ its number of vertices, $|E| = m + b - 1$ its number of edges; thus $b = \beta_1(|G|)$, the first Betti number of G. Let K be a subdivision of G on $n + c - 1$ vertices. Then we have: the exterior algebraic shifting $\Delta^{ex}(K)$ is the same over any field and is a homology lex-segment if and only if it equals

$$\Delta(|G|, n + c - 1) := [n + c - 1] \cup \left\{ A \in \tbinom{[n]}{2} \colon A < \{2, 3 + b\} \right\}. \tag{1}$$

Here and throughout the rest of the article the order $\leq$ stands for the lexicographic order.

Let us denote by (N_g) S_g the (non) orientable closed connected surface with Euler genus g, i.e., g is double the standard genus for orientable surfaces. If K is a triangulation of S_g (N_g) on $n \geq 4 + 3g$ vertices, then its exterior algebraic shifting $\Delta^{ex}(K)$ $(\Delta_2^{ex}(K))$ over a field of characteristic zero (two) is a homology lex-segment if it equals

$$\begin{aligned} \Delta_2(N_g, n) = \Delta(S_g, n) := [n] \cup \left\{ A \in \tbinom{[n]}{2} \colon A < \{4, 5 + 3g\} \right\} \\ \cup \left\{ A \in \tbinom{[n]}{3} \colon A < \{1, 4, 5 + 2g\} \right\} \cup \{\{2, 3, 4\}\}. \end{aligned} \tag{2}$$

On the other hand, if K is a triangulation of N_g, the exterior algebraic shifting $\Delta_0^{ex}(K)$ over a field of characteristic zero is a homology lex-segment if it equals

$$\Delta_0(N_g, n) := [n] \cup \left\{ A \in \tbinom{[n]}{2} \colon A < \{4, 5 + 3g\} \right\} \cup \left\{ A \in \tbinom{[n]}{3} \colon A < \{1, 4, 6 + 2g\} \right\}. \tag{3}$$

We will avoid explicitly mentioning the characteristic of the field except where it is necessary.

The following universality result is our main technical result, of independent interest. It says that for *every* fixed surface triangulation K, all fine enough Delaunay triangulations of same surface do edge contract to K. Formally:

▶ **Theorem 1.1** (Universality for edge contractions). *Let S be a closed connected Riemannian surface and K a triangulation of S. Then, there exists $\rho > 0$ small enough such that if $P \subseteq S$ is a ρ-dense point set, locally in general position, then there exists a sequence of edge contractions from $\mathrm{Del}(P)$ to K such that each intermediate complex triangulates S.*

Being *locally in general position* here means that within every small neighborhood, whose size depends on S, no point of P is on a geodesic between other two points of P and no four of its points lie on a circle. Theorem 1.1 is used to prove the following asymptotic behavior of a random Delaunay triangulation, namely concentration for the exterior algebraic shifting.

▶ **Theorem 1.2** (Concentration of exterior algebraic shifting for Random Delaunay on surfaces). *Let S be a closed connected Riemannian surface. Then, for the random Delaunay model on S there holds:*

- *If S is orientable then, a.a.s., the exterior algebraic shifting satisfies $\Delta^{ex}(\mathrm{Del}(P_n)) = \Delta(S, n)$.*
- *If S is nonorientable and $p \in \{0, 2\}$ then, a.a.s., the exterior algebraic shifting over any fixed field of characteristic p satisfies $\Delta_p^{ex}(\mathrm{Del}(P_n)) = \Delta_p(S, n)$, where Δ_p stands for the exterior algebraic shifting operation over a field of characteristic p.*

Area-rigidity, a generalization of graph-rigidity [1, 2], is concerned with the infinitesimal version of the following question: given a simplicial complex with its vertices embedded in $\mathbb{R}^2$, is there a non-trivial continuous motion of its vertex set that preserves the area of each 2-face. Both area and graph-rigidity admit a characterization in terms of algebraic shifting [21, 6]. While the graph of every surface triangulation is 3-rigid [12], the area-rigidity of its 2-skeleton is a conjecture [6, Conj. 5.1] known to hold for surfaces of small genus [6, Cor. 1.3]. If true, the barycentric subdivision of a surface triangulation would be a homology lex-segment, see Conj. 6.2. Combining the algebraic shifting characterization of area-rigidity [6, Thm.1.2 and Claim 2.2] and Theorem 1.2 we obtain an affirmative answer to the area-rigidity conjecture for surfaces [6, Conj. 5.1] for Delaunay triangulations in the asymptotic regime.

▶ **Corollary 1.3.** *Let S be a closed connected Riemannian surface. Then, a.a.s. $\mathrm{Del}(P_n)$ is area-rigid.*

For uniform random triangulations of a one dimensional compact topological space we show the following concentration for the exterior algebraic shifting.

▶ **Theorem 1.4** (Concentration of exterior algebraic shifting for uniform triangulations in dimension 1). *Let G be a finite graph. Then, a.a.s., the exterior algebraic shifting over any fixed field satisfies $\Delta^{ex}(U_n(|G|)) = \Delta(|G|, n)$.*

The probabilistic conclusion in this theorem can not be upgraded to a deterministic one:

▶ **Example 1.5.** Let G be the graph on 5 vertices containing K_4 and 7 edges, denote by v its vertex of degree one, and by u the neighbor of v. Let R be the n vertex refinement of G obtained by subdividing the edge vu by $n-5$ new vertices. Then K_4 is a subgraph of R hence the edge $\{3, 4\} \in \Delta^{ex}(R)$. But by Theorem 1.4 a.a.s. $\{3, 4\} \notin \Delta^{ex}(H)$ for a uniform n vertex refinement H of G.

Theorems 1.4 and 1.2 lead to the following natural conjecture.

▶ **Conjecture 1.6.** *For every fixed Euler genus g, a.a.s., the exterior algebraic shifting over any fixed field satisfies $\Delta^{ex}(U_n(S_g)) = \Delta(S_g, n)$.*

Outline. In Section 2 we provide preliminaries on exterior algebraic shifting, homology lex-segments, and Delaunay triangulations; in Section 3 we prove Theorem 1.1; in Section 4 we prove Theorem 1.2; in Section 5 we prove Theorem 1.4; in Section 6 we discuss concentration for triangulations of other spaces.

2 Preliminaries

2.1 Algebraic shifting

We recall Kalai's definition [16]. Let K be a simplicial complex with vertex set N, $\mathbb{F}$ be an field extension of a base field $\mathbb{F}'$ of transcendence degree at least n^2, and consider the exterior algebra mod the ideal generated by non-faces $\bigwedge K = \bigwedge \mathbb{F}^N / \operatorname{span}\{e_A \colon A \notin K\}$, called the *exterior face ring of* K. Here the e_i's form the standard basis of $\mathbb{F}^N$, and e_A stands for the exterior product $e_{a_1} \wedge \ldots \wedge e_{a_k}$ where $A = \{a_1 < \ldots < a_k\}$. Take $(f_i)_{i \in N}$ a generic change of basis of $\mathbb{F}^N$, namely all n^2 entries of the transition matrix are algebraically independent over the base field $\mathbb{F}'$. Then, the *exterior algebraic shifting of* K is the simplicial complex given by

$$\Delta^{ex}(K) = \{B \subseteq N \colon \overline{f_B} \notin \operatorname{span}_{\mathbb{F}}\{\overline{f_A} \colon A < B\}\},$$

where $f_A = f_{a_1} \wedge \ldots \wedge f_{a_k}$ and its bar $\overline{f_A}$ denotes its image in $\bigwedge K$. The simplest such complex is obtained when all initial sets are in the exterior algebraic shifting. When taking into account Betti numbers this leads to the construction of homology lex-segment complexes as follows. Let $\mathbf{f} = (f_0, f_1, \dots)$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots)$ be two non-negative integer vectors and set $n = f_0$. Let $I([n], r, m)$ denote the first m elements in $\binom{[n]}{r+1}$ in lex order. Similarly to [3] for $r \geq 0$ we set $\chi_{r-1} = \sum_{j \geq r} (-1)^{j-r}(f_j - \beta_j)$ and consider the families of sets $E = \bigcup_{r \geq 0} I([2, n], r, \chi_r + \beta_r)$ and $C = \bigcup_{r \geq 0} I([2, n], r, \chi_r)$ and finally set $K_{\mathbf{f},\boldsymbol{\beta}} = (1 * C) \cup E$ to be the cone over C with apex 1, union E. A simplicial complex K is a *homology lex-segment complex* if $K = K_{f(K),\beta(K)}$. Next, we describe pairs $(\mathbf{f}, \boldsymbol{\beta})$ giving rise to homology lex-segment complexes. For it, let $\partial_r(n, m)$ denote the minimum m' such that $I([n], r-1, m')$ contains $\partial I([n], r, m) = \{A \in \binom{[n]}{r} \colon A \subseteq B, B \in I([n], r, m)\}$.

▶ **Lemma 2.1** ([5, Lemma A.3])**.** *Let* $\mathbf{f}$, $\boldsymbol{\beta}$ *be two non-negative integer vectors. Then,* $K_{\mathbf{f},\boldsymbol{\beta}}$ *is a simplicial complex with* $f(K_{\mathbf{f},\boldsymbol{\beta}}) = \mathbf{f}$ *and* $\beta(K_{\mathbf{f},\boldsymbol{\beta}}) = \boldsymbol{\beta}$ *if and only if* $\chi_{-1} = 1$ *and* $\partial_r(n-1, \chi_r + \beta_r) \leq \chi_{r-1}$ *for* $r \geq 1$*, where* $n = f_0$*.*

Now, the descriptions of homology lex-segment complexes given in (1)–(3) follows, see also [5, Appendix]. The following corollary is a consequence of [24, Theorem 4.6].

▶ **Corollary 2.2.** *Let* K *and* K' *be triangulations of surfaces* S *and* S' *resp., intersecting in a single 2-face* B*, i.e.,* $K \cap K' = \langle B \rangle$*. Then,* $(K \cup K') \setminus \{B\}$ *is a triangulation of the connected-sum surface* $(S \cup S) \setminus B$*. Moreover, if* $\Delta^{ex}(K)$ *and* $\Delta^{ex}(K')$ *are homology lex-segment complexes then* $\Delta^{ex}((K \cup K') \setminus \{B\})$ *is a homology lex-segment complex as well.*

▶ **Corollary 2.3.** *Let* S *be a closed connected surface with Euler genus* g*, then it admits a triangulation whose exterior algebraic shifting is a homology lex-segment complex.*

Proof. By [19, Theorem 1.2] there is a triangulation of the torus on 10 vertices and a triangulation of the projective plane on 7 vertices whose exterior algebraic shifting are homology lex-segments, see Figure 1(2-3). By Corollary 2.2 attaching iteratively g copies of this triangulation along 2-faces we obtain a triangulation of S_g, or N_g, that is a homology lex-segment complex. ◀

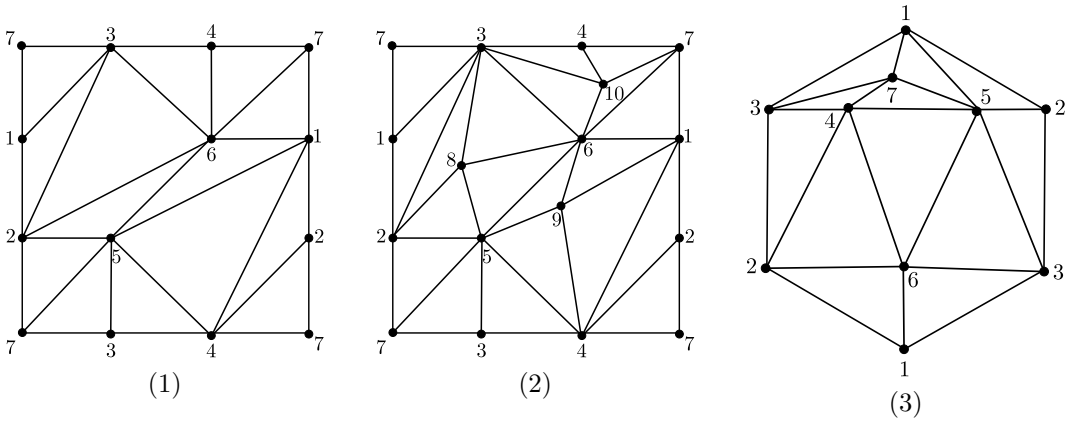


Figure 1 (1) An irreducible triangulation of the torus on 7 vertices. (2) Triangulation of the torus on 10 vertices whose exterior algebraic shifting is a homology lex-segment. (3) Triangulation of the projective plane on 7 whose exterior algebraic shifting is a homology lex-segment over fields with characteristic 0 and 2.

Edge contraction and vertex-split. Let K be a simplicial complex and $e = \{u, v\} \in K$. We say that K' is *obtained from K by contracting the edge e* if K' is obtained from K by replacing every occurrence of the vertex u with the vertex v (and removing duplicated faces). The inverse operation of an edge contraction is called a *vertex-split*. In the case that K is a surface triangulation, an edge contractions results in a triangulation of the same surface if and only if the contracted edge is not part of a missing triangle, namely a triangle not in K whose boundary is contained in K; such edge is called *contractible*. The triangulation of the torus in Figure 1(1) is obtained from Figure 1(2) by means of three edge contractions. Namely, $\{2, 8\}, \{4, 10\}$ and $\{1, 9\}$. Performing a vertex-split on a simplicial complex can only make its exterior algebraic shifting simpler. To state this precisely we set $\operatorname{Tail}(K, A) = \{B \in K \colon A \leq B\}$.

▶ **Proposition 2.4** ([6, 19]). *Let K be a surface triangulation on n vertices and K' obtained from K by a vertex-split. If $\{1, 3, n\} \in \Delta^{ex}(K)$, then $\{1, 3, n+1\} \in \Delta^{ex}(K')$. Moreover, for $m \geq 3$ we have that*

$$|\operatorname{Tail}(\Delta^{ex}(K), \{m+1, m+2\})| \geq |\operatorname{Tail}(\Delta^{ex}(K'), \{m+1, m+2\})|, \text{ and}$$
$$|\operatorname{Tail}(\Delta^{ex}(K), \{1, m+1, m+2\})| \geq |\operatorname{Tail}(\Delta^{ex}(K'), \{1, m+1, m+2\})|.$$

▶ **Corollary 2.5.** *Let K be a surface triangulation and K' obtained from K by a vertex-split. If $\Delta^{ex}(K)$ is a homology lex-segment complex, then $\Delta^{ex}(K')$ is as well.*

Proof. On the one hand, Proposition 2.4 implies that

$$|\operatorname{Tail}(\Delta^{ex}(K'), \{5, 6\})| \leq |\operatorname{Tail}(\Delta^{ex}(K), \{5, 6\})| = 0.$$

On the other hand, Proposition 2.4 also says that $\{1, 3, n'\} \in \Delta^{ex}(K')$, where n' is the number of vertices in K'. Then, the tails $\operatorname{Tail}(\Delta^{ex}(K'), \{4, 5\})$ and $\operatorname{Tail}(\Delta^{ex}(K'), \{1, 4, 5\})$ are totally ordered sets of same sizes as the corresponding tails w.r.t. K, and hence equal to the corresponding tails w.r.t. K (these sizes are determined solely by the Euler genus of the surface). To conclude, $\Delta^{ex}(K')$ is a homology lex-segment complex. ◀

The following lemma and its proof are similar to [19, Lemma 4.10] and [28].

▶ **Lemma 2.6.** *Let K be a triangulation of a disc with at least one edge e adjacent to at least one interior vertex. Then, there exists a contractible edge incident to at least one interior vertex. Moreover, if both endpoints of e are interior then there exists a contractible edge incident to two interior vertices.*

2.2 The Delaunay complex

Riemannian surfaces. An *embedding* of a simplicial complex K into a surface S is a continuous injective map from its geometric realization into S. We will denote by $|K|$ the image of the embedding of K in S. If this map is a homeomorphism then K is said to be a *triangulation* of S. If in addition S has a smooth structure then an embedding is called (piecewise) smooth if restricting it to each edge of K gives rise to a (piecewise) smooth map.

▶ **Theorem 2.7** ([15])**.** *Let K be a simplicial complex and S be a surface with a smooth structure. If K is a triangulation of S then it can be smoothly embedded into S.*

Now, let S be a connected closed Riemannian surface, i.e., compact and without boundary. By the Hopf–Rinow theorem [9] every two points on S are connected by a minimal length geodesic. For $p, q \in S$ let us denote by $d(p, q)$ the distance between p and q given by a minimal length geodesic joining them. For $X \subseteq S$ and $\epsilon > 0$ we will denote $B(X, \epsilon) = \{q \in S \colon d(X, q) < \epsilon\}$ the open neighborhood of a set X of radius ϵ. The *injectivity radius at* $p \in S$, denoted by $\mathrm{r}_{\mathrm{inj}}(p)$, is the largest real number r such that whenever $d(p, q) < r$, then there exists a unique minimal geodesic from p to q. The *injectivity radius of* S is defined as $\mathrm{r}_{\mathrm{inj}}(S) = \inf_{p \in S} \mathrm{r}_{\mathrm{inj}}(p)$. A subset $A \subseteq S$ is said to be *strongly convex* if for every two elements $p, q \in A$ there exists a unique minimal geodesic joining them, this geodesic is contained in A and there is no other geodesic in A joining p and q. The *strong convexity radius at* p is

$$\mathrm{r}_{\mathrm{scv}}(p) = \sup\{r \colon B(p, r') \text{ is strongly convex for every } r' < r\}.$$

The strong convexity radius of S is $\mathrm{r}_{\mathrm{scv}}(S) = \inf_{p \in S} \mathrm{r}_{\mathrm{scv}}(p) > 0$, see [20, 1.9.9].

Tubular neighborhood. For each point $p \in S$ the exponential map at p is the smooth map $\exp_p \colon T_pS \to S$ that assigns for every vector $X \in T_pS$ the point $\gamma(\|X\|)$ where γ is the geodesic starting at p with tangent vector $\gamma'(0) = X/\|X\|$. Here the norm is given by the Riemannian metric on T_pS. Now, let γ be a smooth curve in S and $\epsilon > 0$. At each point $p = \gamma(t)$ let $X \in T_pS$ be an orthonormal vector with respect to $\gamma'(t)$ (there are two choices, one the negative of the other, but both result in the same tubular neighborhood) and set

$$I(p, \epsilon) = \{\exp_p(tX) \colon t \in (-\epsilon, \epsilon)\} \text{ and } I(\gamma, \epsilon) = \bigcup_{p \in \gamma} I(p, \epsilon).$$

There exists $\epsilon > 0$ such that the sets $I(p, \epsilon)$ are all disjoint for $p \in \gamma$ [26, Proposition 7.26]. In this case $I(\gamma, \epsilon)$ is called the *tubular neighborhood of* γ *with radius* ϵ. For a simple curve γ with distinct endpoints a and b we extend its tubular neighborhood by adding a cap on each end point. Concretely, we set $\tilde{I}(\gamma, \epsilon') = B(a, \epsilon') \cup B(b, \epsilon') \cup I(\gamma, \epsilon')$ where $0 < \epsilon' \leq \epsilon$ is chosen to be small enough in order for the caps to be disjoint embedded discs.

Delaunay complex. Let $P \subseteq S$ be a finite subset, *the Delaunay complex of* P *in* S, denoted by $\mathrm{Del}(P)$, is the simplicial complex whose faces are given by subsets $A \subseteq P$ for which there exists an open ball $B(x, \epsilon) \subseteq S$ disjoint from P such that $A \subseteq cl(B(x, \epsilon))$ [8]. Although $\mathrm{Del}(P)$ always defines a simplicial complex, its geometric realization is in general not homeomorphic

to S. In the case of surfaces a density condition suffices to ensure that for a subset $P \subseteq S$ locally in general position, $\mathrm{Del}(P)$ is a smoothly embedded triangulation of S. We make this requirement precise now. Set $\mathrm{r}_{\mathrm{Del}}(S) = \min\{\mathrm{r}_{\mathrm{inj}}(S)/6, \mathrm{r}_{\mathrm{scv}}(S)\}$. A subset $P \subseteq S$ is *locally in general position* if no point is on the minimal geodesic between other two points at distance less than $\mathrm{r}_{\mathrm{Del}}(S)$, and no four points are simultaneously on the boundary of a ball of radius less than $\mathrm{r}_{\mathrm{Del}}(S)$. A set of points $P \subseteq S$ is *ρ-dense* if there is at least one point of P in any ball in S of radius greater or equal to ρ. The existence of a Delaunay triangulation is given by the following theorem, see also [10, Theorem 2].

▶ **Theorem 2.8** ([22]). *Let S be a closed connected Riemannian surface and $\rho \leq \mathrm{r}_{\mathrm{Del}}(S)$. If $P \subseteq S$ is a ρ-dense finite subset locally in general position, then $\mathrm{Del}(P)$ is a geodesically embedded triangulation of S.*

To control the behaviour of Delaunay edges we will repeatedly use the following bound on their length.

▶ **Lemma 2.9.** *Let $P \subseteq S$ be a ρ-dense finite subset such that $\mathrm{Del}(P)$ is a geodesically embedded triangulation of S. Then, for each edge $e \in \mathrm{Del}(P)$ its embedding γ_e has length strictly less than 2ρ.*

Proof. Let $B(x, r)$ be a ball witnessing that $e = \{u, v\}$ is a edge. Then, $r < \rho$ since otherwise it contains $B(x, \rho)$ whose intersection with P is empty contradicting the ρ-dense assumption. Then, $\mathrm{length}(\gamma_e) = d(u, v) \leq d(u, x) + d(x, v) = 2r < 2\rho$ as desired. ◀

3 The refinement of a triangulation

In this section we prove our universality Theorem 1.1. The proof of this theorem proceeds in two steps. The first step, Proposition 3.10, shows that for $\rho > 0$ small enough, a ρ-dense Delaunay triangulation $\mathrm{Del}(P)$ of a surface S edge contracts to a subdivision the original triangulation K of S. The second step, Proposition 3.11, verifies that the edges of a subdivision of K can be contracted in order to reach K. The following is the definition of a subdivision we require, see [23, Sect. 15].

▶ **Definition 3.1.** *Let S be a surface and K, K' be two simplicial complexes embedded in S. The embedding $|K'|$* is a subdivision of *$|K|$ if the embedding of every face of K' is contained in the embedding of some face of K, and the embedding of every face of K is given by a union of the embeddings of finitely many faces from K'. For a face $A \in K$ we denote by K'_A the subcomplex of K' whose realization $|K'_A|$ is a subdivision of $|A| \subseteq S$.*

The following definitions and lemmas pave the way to Proposition 3.10, where we contract edges of $\mathrm{Del}(P)$, locally in small balls around the vertices of K, to reach a subdivision of K; see Figures 2 and 3.

Recall that for an embedded simplicial complex K, the *open star of a face $A \in K$* is the open subset $st^\circ(A, K) \subseteq |K|$ given by the union of the relative interiors of the faces containing A, including A itself. The *closed star* $st(A, K)$ is the closed set $\mathrm{cl}\, st^\circ(A, K)$ or equivalently it is the union of the closures of all the faces containing A [23].

▶ **Definition 3.2.** *Let K be a simplicial complex embedded in S. A family of closed discs with smooth boundaries $\mathcal{B}_0 = \{B_v \subseteq B'_v \subseteq B''_v : v \in K_0\}$ is a* layered vertex cover *if* (1) *$B_v \subseteq \mathrm{int}\, B'_v$ and $B'_v \subseteq \mathrm{int}\, B''_v$,* (2) *$B''_v \cap B''_u = \emptyset$ for every pair of distinct vertices $v, u \in K_0$,* (3) *$v \in B_v \subseteq B'_v \subseteq B''_v \subseteq st^\circ(v, K)$.*

The following intuitive and simple lemma guarantees that a layered vertex cover exists.

▶ **Lemma 3.3.** *Let S be a Riemannian surface, K an embedded triangulation of S. Then, there exists $\epsilon > 0$ such that $\{B(v,\epsilon) \subseteq B(v,2\epsilon) \subseteq B(v,3\epsilon) \colon v \in K_0\}$ is a layered vertex cover.*

▶ **Definition 3.4.** *Let K be a simplicial complex piecewise smoothly embedded in S and $\mathcal{B}_0$ a layered vertex cover. The embedding is* transversal with respect to $\mathcal{B}_0$ *if*

1. *for each edge $e = \{u,v\}$ its embedding γ_e intersects ∂B_v exactly once, at a point $a_{e,v}$,*
2. *for each edge $e = \{u,v\}$ its embedding γ_e is given by a concatenation $\gamma_{e,u} \circ \widetilde{\gamma_e} \circ \gamma_{e,v}$ where $\gamma_{e,u}$ is a smooth curve in B_u connecting $a_{e,u}$ and u, $\gamma_{e,v}$ is smooth curve in B_v connecting v to $a_{e,v}$ and $\widetilde{\gamma_e}$ is a smooth curve connecting $a_{e,v}$ and $a_{e,u}$.*

The following lemma guarantees the existence of a transversal embedding.

▶ **Lemma 3.5.** *Let K be a triangulation of S, then K admits a piecewise smooth embedding in S that is transversal with respect to some layered vertex cover.*

▶ **Definition 3.6.** *Let K be a piecewise smooth embedding into S that is transversal with respect to the layered vertex cover $\mathcal{B}_0$. A family of closed discs $\mathcal{B}_1 = \{B_e \subseteq S \colon e \in K_1\}$ is an* edge cover *if it satisfies the following four conditions:* (1) $\widetilde{\gamma_e} \subseteq \operatorname{int} B_e$, (2) $B_e \subseteq st^\circ(e,K)$, (3) $B_e \cap B_{e'} = \emptyset$ *for every pair of distinct edges $e, e' \in K_1$,* (4) $B_e \cap B''_v = \emptyset$ *if $v \notin e$.*

▶ **Lemma 3.7.** *Let K be a simplicial complex piecewise smoothly embedded in S that is transversal with respect to the layered vertex cover $\mathcal{B}_0$. Then, there exists $\delta > 0$ such that $\{B(\widetilde{\gamma_e},\delta) \colon e \in K_1\}$ is an edge cover.*

Proof. The curve $\widetilde{\gamma_e}$ is contained in the interior of the tubular neighborhood with caps $\operatorname{int} \widetilde{I}(\widetilde{\gamma_e})$ and consequently by setting $0 < \delta \leq d(\widetilde{\gamma_e}, \partial \widetilde{I}(\gamma_e))$ we guarantee that the resulting set is an embedded disc. Property (1) holds by construction, and small enough δ clearly guarantees (2) as well. Since $\widetilde{\gamma_e}$ and $\widetilde{\gamma_{e'}}$ are disjoint for every pair of distinct edges $e, e' \in K$, then by setting $\delta \leq d(\widetilde{\gamma_e}, \widetilde{\gamma_{e'}})/2$ property (3) holds. Finally, as $\gamma_e \cap B''_v = \emptyset$ for $v \notin e$ since $B''_v \subseteq st^\circ(v,K)$ then by setting $0 < \delta \leq d(\widetilde{\gamma_e}, B''_v)$ also property (4) is satisfied. ◀

▶ **Lemma 3.8.** *Let S be a closed connected Riemannian surface, let $X \subseteq D \subseteq S$ be a pair of discs in S and $P \subseteq S$ a ρ-dense finite set such that $\operatorname{Del}(P)$ is a triangulation of S. If $2\rho < \min\{d(X, \partial D), \mathrm{r}_{\mathrm{scv}}(S)\}$, then the restriction of $\operatorname{Del}(P)$ to the triangles all whose vertices are in D, denoted by Del_D, contains a unique maximal disc subcomplex $\operatorname{Del}_{D,X}$ containing X. Further, all the boundary vertices of $\operatorname{Del}_{D,X}$ lie outside of X.*

Proof. As $2\rho < \mathrm{r}_{\mathrm{scv}}(S)$, every cycle on the edges of Del_D is fully contained in the disc D, hence so is its inner connected component disc C. Since $\operatorname{Del}(P)$ is a triangulation of S, it has a subcomplex that triangulates C, and its triangles are present in Del_D because they are contained in D. Thus, Del_D is a simply connected planar pure 2-dimensional complex, hence a *cacti*, see e.g. [7], namely the union of maximal simplicial discs, every two of them are either disjoint or intersect in a single vertex, and the bipartite graph whose vertices correspond to the set A of discs and the set B of their intersection points and whose edges ab correspond to incidences $b \in a$ for $(a,b) \in A \times B$, is a forest.

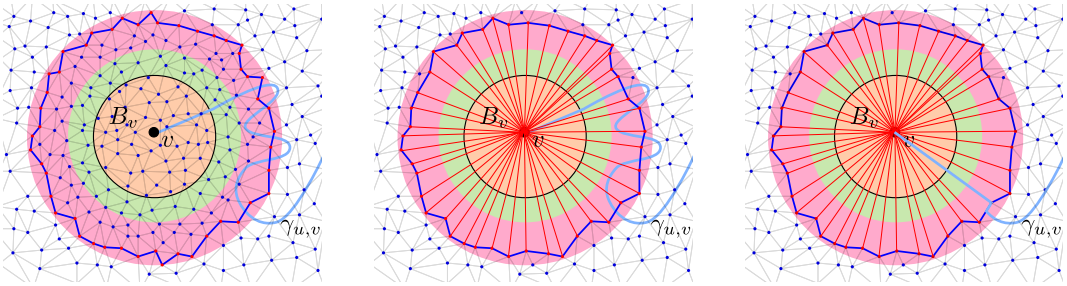
Denote by $\operatorname{Del}_{D,X}$ the maximal disc in the decomposition that contains X. It is left to show that $\operatorname{Del}_{D,X}$ is well defined. Indeed, for every point $x \in X$, each vertex in a triangle A in $\operatorname{Del}(P)$ containing x is of distance at most 2ρ from x. Since $2\rho < \mathrm{r}_{\mathrm{scv}}$ the embedding of A is contained in $B(x,2\rho)$ since this last one is strongly convex. Moreover, given that $2\rho < d(x,\partial D)$ the ball $B(x,2\rho)$, and consequently A as well, is contained in D. Thus, the

triangle A is in Del_D, and consequently there exists a unique disc component $\mathrm{Del}_{D,X}$ in the cacti that contains X. Further, as $2\rho < d(X, \partial D)$, for a vertex $v \in \mathrm{Del}(P)$ lying in X, all its neighbors are in D, hence such v can not be on the boundary of $\mathrm{Del}_{D,X}$. ◀

▶ **Lemma 3.9.** *Let $B \subseteq B' \subseteq B'' \subseteq S$ closed discs with smooth boundary such that $B \subseteq \operatorname{int} B'$, $B' \subseteq \operatorname{int} B''$, $\rho > 0$ and $P \subseteq S$ a ρ-dense finite point set such that $\mathrm{Del}(P)$ is a triangulation of S. If $2\rho < \min\{d(B, \partial B'), d(B', \partial B''), \mathrm{r_{scv}}\}$, then the vertex set of $\partial\,\mathrm{Del}_{B'',B}$ is contained in $B'' \setminus B'$ and every edge between boundary vertices of $\mathrm{Del}_{B'',B}$ is disjoint from B.*

Proof. As $B \subseteq B'$, the maximal discs $\mathrm{Del}_{B'',B}$ and $\mathrm{Del}_{B'',B'}$ from Lemma 3.8 satisfy, by definition, that they are equal, and their boundary vertices lie in $B'' \setminus B'$.

Now, let $e = \{u, v\}$ be a Delaunay edge with endpoints in $B'' \setminus B'$. Then, $\gamma_e \cap B = \emptyset$ since $d(B, \partial B') > 2\rho$ and all edges have length at most 2ρ. ◀



(1) (2) (3)

Figure 2 (1) The Delaunay triangulation in a neighborhood of the vertex v, where layered vertex cover of v is depicted. The vertices of $\partial\,\mathrm{Del}_{B''_v,B_v}$ (marked in red) and its edges (marked in blue) are disjoint from B_v. The embedding $\gamma_{u,v}$ is shown in light blue. (2) The triangulation obtained after contracting edges inside D_v. The resulting simplicial complex is a cone. Note that the vertices on the boundary of D_v form a strict subset of those on the boundary of $\mathrm{Del}_{B''_v,B_v}$. (3) The embedding of the edge $\gamma_{u,v}$ is modified: after intersecting ∂D_v, it continues along the shortest path to v.

▶ **Proposition 3.10.** *Let S be a closed Riemannian manifold and K a triangulation of S. There exist a piecewise smooth embedding of K in S and a density $\rho > 0$ such that the following holds. If $P \subseteq S$ is a ρ-dense finite set locally in general position, then $\mathrm{Del}(P)$ admits a sequence of edge contractions reaching a subdivision of K.*

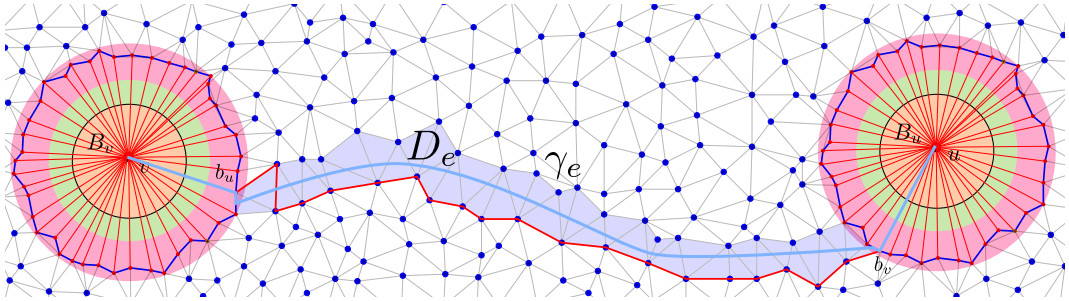


Figure 3 Edge approximation in a Delaunay triangulation inside the edge neighborhood Del_e.

Proof. First, by Lemma 3.5 the triangulation K admits a piecewise smooth embedding in S that is transversal with respect to some layered vertex cover $\mathcal{B}_0$. In addition, let $\mathcal{B}_1$ be the edge cover guaranteed by Lemma 3.7. Finally, set $\rho > 0$ small enough, to be specified below, that depends only on S, on the embedding of K, on the layered vertex cover $\mathcal{B}_0$ and on the edge cover $\mathcal{B}_1$.

Let $\rho < \mathrm{r}_{\mathrm{Del}}(S)$ and $P \subseteq S$ a ρ-dense finite subset locally in general position. Then, Theorem 2.8 implies that the simplicial complex $\mathrm{Del}(P)$ is a triangulation of S.

Call an edge in a triangulated disc *diagonal* if both its vertices belong to the boundary of the disc. Now, let $\mathrm{Del}_{B''_v,B_v}$ be the disc guaranteed by Lemma 3.8, see Figure 2(1). We proceed to select a subcomplex of $\mathrm{Del}_{B''_v,B_v}$ without diagonal edges while still containing B_v in its interior. For it, let e be a diagonal edge in $\mathrm{Del}_{B''_v,B_v}$. Then e splits this complex into two connected components both of which are discs. By taking $2\rho < \min\{d(B_v, \partial B'_v), d(B'_v, \partial B''_v), \mathrm{r}_{\mathrm{scv}}(S)\}$, Lemma 3.9 guarantees that the embedding γ_e does not intersect B_v. We iterate this procedure with the component having B_v in its interior. The resulting triangulated disc D has no diagonal edges and contains B_v in its interior. In D we proceed to iteratively contract edges whose both endpoints are in the interior of D, see Lemma 2.6. The resulting complex D_v is a star with boundary ∂D and we identify the apex vertex with v, see Figure 2(2). Let K' denote the result of applying this sequence of edge contractions to $\mathrm{Del}(P)$ for every vertex $v \in K_0$.

Next we modify the embedding of K so that its edges are realized on the 1-skeleton of K'. For it, let $e = \{u, v\} \in K_1$ and $\mathrm{Del}_{B_e,\widetilde{\gamma_e}} \subseteq \mathrm{Del}(P)$ given by Lemma 3.8[1], where we have applied the lemma to a small enough neighborhood of $\widetilde{\gamma_e}$. Since $\widetilde{\gamma_e}$ intersects ∂D_u and ∂D_v so does $\mathrm{Del}_{B_e,\widetilde{\gamma_e}}$. Let γ'_e be a shortest path in the graph metric contained in the 1-skeleton of $\mathrm{Del}_{B_e,\widetilde{\gamma_e}}$ between ∂D_u and ∂D_v with endpoints $b_{e,u}$ and $b_{e,v}$ respectively, see Figure 3. We need to make sure that for $e = vu$ and $e'' = vu''$ the vertices $b_{e,v}$ and $b_{e'',v}$ in D_v are distinct. Indeed, by taking $4\rho < d(\gamma_e \cap (B''_v \setminus B_v), \gamma_{e''} \cap (B''_v \setminus B_v))$ for every two edges e, e'' both containing v as a vertex, we guarantee that $b_{e,v}$ and $b_{e'',v}$ are distinct. Finally, replace the embedding γ_e by the concatenation $\gamma'_{e,u} \circ \gamma'_e \circ \gamma'_{e,v}$ where $\gamma'_{e,u}$ is the unique edge in D_u from $b_{e,u}$ to u and similarly $\gamma'_{e,v}$ is the unique edge in D_v from $b_{e,v}$ to v, see Figures 2(3) and 3. Since the cyclic order at every vertex of K is preserved, the embedded simplicial complex coincides with K (combinatorially) and it is subdivided by $|K'|$. ◀

▶ **Proposition 3.11.** *Let K, K' be two embedded triangulations of a surface S such that $|K'|$ is a subdivision of $|K|$. Then, there exists a sequence of edge contractions from K' to K.*

Proof. We proceed by induction on the number of edges of K'. We split the analysis into two cases, in each case we perform an edge contraction to reduce the number of edges and proceed by induction. If the vertex sets $K_0 = K'_0$ then the simplicial complexes coincide. Let us assume then that K' has more vertices and consequently there exists a face $A \in K$ such that its subdivision K'_A has at least one interior vertex. We split the analysis into two cases, either there exists such a triangle A, or else A must be an edge.

Case 1: If A is a triangle, we let K'_A be the subdivision of a 2-face with an interior vertex. Then, by Lemma 2.6 there exists a contractible edge incident to an interior vertex.

[1] The definition of $\mathrm{Del}_{B,X}$ applies to every connected subspace X contained in the interior of a disc B. Indeed, as X is far enough from the boundary of B then an open neighborhood of X is contained in $\mathrm{Del}(P) \cap B$, hence X is contained in a maximal disc component – this is $\mathrm{Del}_{B,X}$.

Case 2: Suppose that there are no more 2-faces of K whose subdivision has an interior vertex. Let K'_e be the subdivision of an edge $e \in K$ with a vertex in its relative interior and let $f = \{a, b\} \in K'_e$ an edge. If f is contractible, then we contract it and proceed by induction. Otherwise, f is part of a missing triangle $T = \{a, b, c\}$ in K', hence c must be in K'_e as we excluded Case 1.

The boundary of T is contained in the disc D formed by the embedded two triangles in K that contain e. Thus, the boundary of T bounds a disc D_T realized inside $|D|$ which is a subcomplex of K' with an interior vertex; all vertices of D_T belong to K'_e. As K'_e has finitely many vertices, proceeding in this manner for an edge f' in $K'_e \cap D_T$ and so on, we finally find a contractible edge in K'_e, similar to the proof of Lemma 2.6. ◀

4 Concentration of exterior algebraic shifting for Random Delaunay

Let ν be a volume measure on S and U_S be the random variable on S uniformly distributed with respect to ν, i.e., for a ν-measurable subset $V \subseteq S$ we have that $\mathbb{P}(U_S \in V) = \nu(V)/\nu(S)$. For $n \in \mathbb{N}$ let P_n denote the random variable of picking n (unlabelled) points from S independently uniformly at random according to ν. Equivalently, we can consider the stochastic process $(P_n)_{n\in\mathbb{N}}$ where at each step we pick a new point in S uniformly at random and independent from the previous choices. We are interested in sampling points that are locally in general position (g.p. for short), i.e., (1) no point is on the minimal geodesic between other two points at distance less than $\mathrm{r}_{\mathrm{Del}}(S)$, and (2) no four points are simultaneously on the boundary of a ball of radius less than $\mathrm{r}_{\mathrm{Del}}(S)$.

▶ **Lemma 4.1.** *Let S be a closed connected Riemannian surface, $\rho > 0$ fixed, and P_n as above. Then, P_n is almost-surely locally in g.p for every $n \in \mathbb{N}$, and a.a.s. ρ–dense.*

Proof. For the first part we proceed by induction on n. Denote by E_n the event that P_n is locally in g.p. Observe that $\mathbb{P}(E_n) = \mathbb{P}(E_n|E_{n-1})\mathbb{P}(E_{n-1})$ and that $\mathbb{P}(E_{n-1}) = 1$ by induction. Conditioned on E_{n-1}, the probability of E_n is given by the probability of picking a point that is not in the union of the sets determined by the complements of the conditions (1) and (2) in the definition of g.p. above. Since these complements have ν-measure 0 in S and there are finitely many of them, the conclusion follows.

For the second part let $\{B(x, \rho/2) \colon x \in I\}$ be a finite open cover of S, which exists since S is compact. Then, it is enough to guarantee that $P_n \cap B(x, \rho/2) \neq \emptyset$ for every $x \in I$. Indeed, if this is true, then for every $y \in S$ there exists $x \in I$ such that $y \in B(x, \rho/2)$ and $p \in P \cap B(x, \rho/2)$. Then, $d(y, p) \leq d(y, x) + d(x, p) < \rho$ as claimed. It follows that

$$\begin{aligned}\mathbb{P}(P_n \text{ is } \rho\text{–dense}) &\geq \mathbb{P}(\forall x \in I, P_n \cap B(x, \rho/2) \neq \emptyset) \geq 1 - \sum_{x\in I} \mathbb{P}(P_n \cap B(x, \rho/2) = \emptyset) \\ &\geq 1 - |I|(1 - \mu)^n,\end{aligned}$$

where $\mu = \min_{x\in I} \nu(B(x, \rho/2))/\nu(S) > 0$. The conclusion now follows by taking $n \to \infty$. ◀

Proof of Theorem 1.2. Let K be a triangulation of S given by Corollary 2.3 and set $\rho > 0$ as required by Theorem 1.1 when applied to K and S. By Lemma 4.1 P_n is a.a.s. ρ–dense and is locally in general position. Then, Theorem 1.1 guarantees that $\mathrm{Del}(P_n)$ admits a sequence of edge contractions reaching K. Since $\Delta^{ex}(K)$ is a homology lex-segment then Corollary 2.5 implies that $\Delta^{ex}(\mathrm{Del}(P_n))$ is a homology lex-segment as well. ◀

5 Concentration of exterior algebraic shifting for the uniform model

One dimensional complexes. Let X be a compact one dimensional topological space. It admits natural triangulations each given by some graph G. Let $\mathrm{sd}(G)$ denote the barycentric subdivision of G, namely the one obtained by introducing a new vertex at the interior of each edge of G, subdividing that edge into two edges. The following proposition holds for exterior algebraic shifting over any field:

▶ **Proposition 5.1.** *Let G be a graph, then $\Delta^{ex}(\mathrm{sd}\, G)$ is a homology lex-segment.*

Proof. First, we claim that $\mathrm{Tail}(\Delta^{ex}(\mathrm{sd}\, G), \{3,4\}) = \emptyset$. Otherwise there exists a connected subgraph H of $\mathrm{sd}\, G$ that is a 2-hypercycle and consequently is 3-edge connected, see [17, Theorem 5.4]. However, this is not possible for if $e = \{u, v\} \in H$ and we assume without loss of generality that $u \in V(\mathrm{sd}\, G) \setminus V(G)$ then the degree of u in H is at most 2. Since algebraic shifting outputs a shifted simplicial complex with identical Betti numbers we conclude that $\Delta^{ex}(\mathrm{sd}\, G) = [n] \bigcup \{A \in \binom{[n-\beta_0]}{2} : A \leq \{2, \beta_1 + 2\}\}$ as wanted. ◀

Proof of Theorem 1.4. First, by Propositions 5.1 and 2.4 with $m = 2$, the event that $\Delta^{ex}(U_n(|G|)) = \Delta(|G|, n)$ contains the event E that every edge of G is subdivided. In addition, since the order in which we subdivide an edge is not important the model $U_n(|G|)$ can be viewed as a balls and bins model where the bins represent the edges of G and the balls represent the new vertices. Therefore, the probability that E does not occur is at most $|E|(1 - 1/|E|)^n$ which tends to 0 as $n \to \infty$. ◀

Two dimensional complexes. We show here that unlike the one-dimensional case, the uniform triangulation of the 2-disc and of the disjoint union of two 2-discs do not have a concentrated exterior algebraic shifting.

▶ **Proposition 5.2.** *Let K be a triangulation of a disc with n interior vertices and m boundary vertices, then*

$$\Delta^{ex}(K) = [n+m] \bigcup \left\{ A \in \binom{[n+m]}{3} : A \leq \{1, 3, 3+n\} \right\}.$$

Proof. Since K is part of a triangulation of a sphere, then $\{1,4,5\} \notin \Delta^{ex}(K)$. Moreover, since the homology of K is trivial, then there is no 2-face whose smallest vertex is strictly larger than 1. Since the 1-skeleton of K is 2-hyperconnected, see [17], then $\{2, n+m\} \in \Delta^{ex}(K)$. Combining this with the fact that K is Cohen–Macaulay, even shellable, its algebraic shifting is pure implying that $\{1, 2, m+n\} \in \Delta^{ex}(K)$ and $\{4,5\} \notin \Delta^{ex}(K)$. Thus, the remaining edges must be of the form $\{3, k\}$, and the remaining triangles of the form $\{1, 3, k\}$, forming initial lex-segments of edges and of triangles by the shiftedness of $\Delta^{ex}(K)$. The claim now follows from the fact that algebraic shifting preserves the f-vector. ◀

▶ **Corollary 5.3.** *Let X be a 2-dimensional disc, then $\Delta^{ex}(U_n(X))$ is not concentrated.*

Proof. By well-known estimates [27, 14] for the number of n-vertex triangulations of the disc with m boundary vertices, it can be derived that for every fixed $m \geq 3$ there exists $\varepsilon_m > 0$ such that $\mathbb{P}(U_n(X) \text{ has } m \text{ boundary vertices}) = \varepsilon_m + o(1)$ as $n \to \infty$. In consequence, the probability that the 2-faces satisfy $\Delta^{ex}(U_n(X)) = \left\{ A \in \binom{[n+m]}{3} : A \leq \{1, 3, 3+n\} \right\}$, for every fixed m, is bounded away from 0 as n grows. ◀

▶ **Lemma 5.4.** *Let K_1, K_2 be the triangulations of two disjoint discs with n_1, n_2 interior vertices and m_1, m_2 boundary vertices respectively. Set $n = n_1 + n_2$ and $m = m_1 + m_2$, then*

$$\Delta^{ex}(K_1 \sqcup K_2) = [n+m] \bigcup \{\{i,j\} : 1 \le i \le 3,\ i \le t \le n+m-i\} \bigcup \left\{ A \in \binom{[n+m-2]}{3} : A \le \{1,3,n-3\} \right\}.$$

Proof. The proof follows by applying Proposition 5.2 and [24, Theorem 4.6] for exterior shifting of a disjoint union of simplicial complexes. ◀

▶ **Corollary 5.5.** *Let $X = X_1 \sqcup X_2$ be two disjoint 2-dimensional discs, then $\Delta^{ex}(U_n(X))$ is neither concentrated nor a homology lex-segment.*

Proof. Let K_1, K_2 and n and m as in Lemma 5.4, then we have that $\{2, n+m-1\} \notin \Delta^{ex}(K_1 \sqcup K_2)$ while $\{1, n+m-1\}, \{3, n-3\} \in \Delta^{ex}(K_1 \sqcup K_2)$, showcasing that the complex is not an homology lex-segment. The lack of concentration follows from Corollary 5.3. ◀

6 Concluding remarks

Conjecture 1.6 states that the exterior shifting of the uniform triangulation $U_n(S_g)$ of a fixed orientable surface is a homology lex-segment. In particular, this implies that $U_n(S_g)$ is a.a.s. K_6-free (as otherwise the shifting contains the edge $\{5,6\}$). To the best of our knowledge, this statement is not known, although [4] shows that a fixed neighborhood around *almost* every vertex is a.a.s. planar. We now suggest an even more far-reaching problem, which arises naturally as a possible approach to extending our proof of Theorem 1.2 to Conjecture 1.6:

▶ **Problem 6.1.** *Fix an Euler genus g. Show that there exists a Riemannian metric for S_g such that for every fixed $\rho > 0$, $U_n(S_g)$ can a.a.s. be embedded in S_g such that every edge has length less than ρ.*

In addition, note that in order to prove the analogous Theorem 1.4 for graphs, we first showed the deterministic statement Proposition 5.1 on their barycentric subdivision. We conjecture that its analog holds for barycentric subdivision of surfaces as well:

▶ **Conjecture 6.2.** *If K is a surface triangulation, then $\Delta^{ex}(\mathrm{sd}(K))$ is a homology lex-segment.*

Conjecture 6.2 follows from the following one, for exterior algebraic shifting over any field:

▶ **Conjecture 6.3** ([6, Conj.5.1] for the case of characteristic zero)**.** *For every triangulation K on n vertices of a connected compact surface without boundary, $\{1,3,n\} \in \Delta^{ex}(K)$.*

Proof of Conjecture 6.2 modulo Conjecture 6.3. Let K triangulate S_g and denote $m = f_0(\mathrm{sd}(K))$. The 1-skeleton of $\mathrm{sd}(K)$ is 3-hyperconnected, see e.g. [25, Thm.3.4.2], and similarly over every field, hence $\{3,m\} \in \Delta^{ex}(\mathrm{sd}(K))_1$. To see that $\{5,6\} \notin \Delta^{ex}(\mathrm{sd}(K))_1$, it is enough to remove from $\mathrm{sd}(K)$ vertices of degree at most 4 one by one until we reach a collection of isolated vertices; see Kalai [17, Lem.4.3(i)] for shifting over a field of characteristic zero, and the same proof holds over every field. First we remove vertices at the barycenter of edges of K (their degree is 4), then we remove vertices at the barycenter of triangles of K (their degree at this stage is 3), to reach the original vertices of K, and no higher dimensional faces. As $\Delta^{ex}(\mathrm{sd}(K))$ is shifted and has the same number of edges as $\mathrm{sd}(K)$, we conclude

that $\Delta^{ex}(\mathrm{sd}(K))_1 = \{A \in \binom{[m]}{2} : A < \{4, 5+3g\}\}$. Assuming Conjecture 6.3, $\{1,3,m\} \in \Delta^{ex}(\mathrm{sd}(K))_2$, and as $\Delta^{ex}(\mathrm{sd}(K))$ is a shifted simplicial complex, not containing the edge $\{5,6\}$ with the same face and Betti numbers as $\mathrm{sd}(K)$ we conclude that $\Delta^{ex}(\mathrm{sd}(K))_2 = \{A \in \binom{[m]}{3} : A < \{1,4,5+2g\}\} \cup \{\{2,3,4\}\}$. ◀

In order to extend Theorem 1.2 to fields of characteristics different from 0 and 2 we only need to show that Corollary 2.3 holds over such fields. We leave this as an open problem.

▶ **Problem 6.4.** *Let K be the triangulation of either the torus or the projective plane appearing in Figure 1(2-3). Then for every prime p, the exterior algebraic shifting of K over a field of characteristic p, $\Delta_p^{ex}(K)$, is a homology lex-segment.*

References

1 L. Asimow and B. Roth. The rigidity of graphs. *Trans. Amer. Math. Soc.*, 245:279–289, 1978. doi:10.2307/1998867.

2 L. Asimow and B. Roth. The rigidity of graphs. II. *J. Math. Anal. Appl.*, 68(1):171–190, 1979. doi:10.1016/0022-247X(79)90108-2.

3 Anders Björner and Gil Kalai. An extended Euler-Poincaré theorem. *Acta Math.*, 161(3-4):279–303, 1988. doi:10.1007/BF02392300.

4 Thomas Budzinski. Multi-ended Markovian triangulations and robust convergence to the UIPT. *Ann. H. Lebesgue*, 5:1235–1259, 2022. doi:10.5802/ahl.149.

5 Denys Bulavka, Eran Nevo, and Yuval Peled. The typical algebraic shifting of a surface. *arXiv preprint arXiv:2509.26525*, 2025.

6 Denys Bulavka, Eran Nevo, and Yuval Peled. Volume rigidity and algebraic shifting. *J. Combin. Theory Ser. B*, 170:189–202, 2025. doi:10.1016/j.jctb.2024.09.002.

7 Ralph L. Cohen and John D. S. Jones. A homotopy theoretic realization of string topology. *Math. Ann.*, 324(4):773–798, 2002. doi:10.1007/s00208-002-0362-0.

8 Boris Delaunay. Sur la sphère vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk SSSR*, 7(793-800), 1934.

9 Manfredo Perdigão do Carmo. *Riemannian geometry*. Mathematics: Theory & Applications. Birkhäuser Boston, Inc., Boston, MA, 1992. doi:10.1007/978-1-4757-2201-7.

10 Ramsay Dyer, Hao Zhang, and Torsten Möller. Surface sampling and the intrinsic Voronoi diagram. *Computer Graphics Forum*, 27(5):1393–1402, 2008. doi:10.1111/j.1467-8659.2008.01279.x.

11 Paul Erdős, Chao Ko, and Richard Rado. Intersection theorems for systems of finite sets. *Quart. J. Math. Oxford Ser. (2)*, 12:313–320, 1961. doi:10.1093/qmath/12.1.313.

12 A. L. Fogelsanger. *The generic rigidity of minimal cycles*. ProQuest LLC, Ann Arbor, MI, 1988. Thesis (Ph.D.)–Cornell University. URL: http://gateway.proquest.com/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqdiss&rft_dat=xri:pqdiss:8821159.

13 Peter Frankl. The shifting technique in extremal set theory. In *Surveys in combinatorics 1987 (New Cross, 1987)*, volume 123 of *London Math. Soc. Lecture Note Ser.*, pages 81–110. Cambridge Univ. Press, Cambridge, 1987.

14 Ian P. Goulden and David M. Jackson. *Combinatorial enumeration*. Dover Publications, Inc., Mineola, NY, 2004.

15 Allen Hatcher. The Kirby torus trick for surfaces. *Enseign. Math.*, 72(1-2):161–174, 2026. doi:10.4171/lem/1096.

16 Gil Kalai. Characterization of f-vectors of families of convex sets in $\mathbb{R}^d$, part I: Necessity of Eckhoff's conditions. *Israel Journal of Mathematics*, 48:175–195, 1984. doi:10.1007/BF02761163.

17 Gil Kalai. Hyperconnectivity of graphs. *Graphs Combin.*, 1(1):65–79, 1985. doi:10.1007/BF02582930.

18 Gil Kalai. Algebraic shifting. In *Computational commutative algebra and combinatorics (Osaka, 1999)*, volume 33 of *Adv. Stud. Pure Math.*, pages 121–163. Math. Soc. Japan, Tokyo, 2002. doi:10.2969/aspm/03310121.

19 Aaron Keehn and Eran Nevo. Exterior shifting of low genus surfaces. *arXiv preprint arXiv:2405.12758*, 2024.

20 Wilhelm P. A. Klingenberg. *Riemannian geometry*, volume 1 of *De Gruyter Studies in Mathematics*. Walter de Gruyter & Co., Berlin, second edition, 1995. doi:10.1515/9783110905120.

21 C. W. Lee. Generalized stress and motions. In *Polytopes: abstract, convex and computational (Scarborough, ON, 1993)*, volume 440 of *NATO Adv. Sci. Inst. Ser. C: Math. Phys. Sci.*, pages 249–271. Kluwer Acad. Publ., Dordrecht, 1994.

22 Gregory Leibon. *Random Delaunay triangulations, the Thurston-Andreev theorem, and metric uniformization*. ProQuest LLC, Ann Arbor, MI, 1999. Thesis (Ph.D.)–University of California, San Diego. URL: http://gateway.proquest.com/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqdiss&rft_dat=xri:pqdiss:9936832.

23 James R. Munkres. *Elements of algebraic topology*. Addison-Wesley Publishing Company, Menlo Park, CA, 1984.

24 Eran Nevo. Algebraic shifting and basic constructions on simplicial complexes. *J. Algebraic Combin.*, 22(4):411–433, 2005. doi:10.1007/s10801-005-4626-0.

25 Eran Nevo. Algebraic shifting and f-vector theory. *PhD thesis, arXiv preprint arXiv:0709.3265*, 2007.

26 Barrett O'Neill. *Semi-Riemannian geometry*, volume 103 of *Pure and Applied Mathematics*. Academic Press, Inc. Harcourt Brace Jovanovich, Publishers, New York, 1983.

27 William T. Tutte. A census of planar triangulations. *Canadian J. Math.*, 14:21–38, 1962. doi:10.4153/CJM-1962-002-9.

28 Walter Whiteley. La division de sommet dans les charpentes isostatiques. *Structural Topology*, 16:23–30, 1990.

Finding a Fair Scoring Function for Top-k Selection: From Hardness to Practice

Guangya Cai
University of Minnesota, Twin Cities, MN, USA

Abstract

We study the problem of finding a fair linear scoring function over (numerical) attributes for top-k selection, ensuring fairness through a proportional representation constraint on the protected group. Existing algorithms do not scale efficiently, particularly in higher dimensions. Our hardness analysis shows that in more than two dimensions, no algorithm is likely to scale efficiently with respect to dataset size, and the computational complexity is likely to grow rapidly with dimensionality. However, the hardness results also provide key insights guiding algorithm design, leading to our two-pronged solution: (1) For small k, our analysis reveals a gap in the hardness barrier. By addressing various engineering challenges, including achieving efficient parallelism, we turn this potential of efficiency into an optimized geometry-based algorithm delivering substantial performance gains. (2) For large k, where the hardness is robust, we employ a practically efficient optimization-based algorithm which, despite being theoretically worse, achieves superior real-world performance. Experimental evaluations on real-world datasets then explore scenarios where worst-case behavior does not manifest, identifying areas critical to practical performance. Our solution achieves speedups of up to several orders of magnitude compared to the state of the art, an efficiency made possible through a tight integration of hardness analysis, algorithm design, practical engineering, and empirical evaluation.

2012 ACM Subject Classification Information systems → Top-k retrieval in databases

Keywords and phrases Fairness, Top-k, Integration

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.26

Related Version *Full Version*: `https://arxiv.org/abs/2503.11575` [11]

Supplementary Material *Software*: `https://github.com/caiguangya/fair-topk`
archived at `swh:1:dir:71bbb9faeffa0e61b7f5f063e08ec7e80d6ca6ec`

Acknowledgements The author would like to acknowledge the Minnesota Supercomputing Institute (MSI) at the University of Minnesota for providing computing facilities.

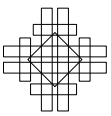
1 Introduction

Selecting a subset of items from a large dataset based on specific criteria is critical for decision-making. Each item is typically assigned a score derived from its attributes, and the top-scoring items are selected. A scoring function determines these scores, typically a linear one that weights and sums the attribute values by importance. The problem of selecting the k highest scoring items from a dataset of n items ($k \leq n$) is called *Top-k Selection*.

Given the rise of automated decision-making software, ensuring fairness in tasks like top-k selection has become crucial. Among different ways of quantifying fairness (e.g., [21, 50, 52]), a common one is based on proportional representation: Each item has categorical attributes (e.g., gender, race, ethnicity), and one or a combination of them is considered *sensitive*. A specific value of this sensitive attribute denotes membership in a minority or historically disadvantaged group, often referred to as a *protected* group. The aim is to select a top-k subset where the protected group's proportion mirrors its proportion in the entire dataset.

Many applications seek top-k selections that are both fair and high-quality, often achieved by applying proportional fairness constraints. Prior work (e.g., [51, 13, 49]) typically applies these constraints after scores are determined, which might result in group-dependent selection


© Guangya Cai;
licensed under Creative Commons License CC-BY 4.0



criteria and thus, potential legal risks (e.g., Ricci vs. DeStefano [42]). A fair scoring function from the outset is thus preferable. Moreover, one may also wish to find such a function within a subset of scoring functions. For instance, one may start with an existing, unfair scoring function and seek a "nearby" fair one, where attribute weights do not vary by a lot. Even knowing that no such nearby function exists can be informative, revealing some negligence in the original design. Consider the following variation of the example from [6]:

▶ **Example 1.** *A college admissions officer designs a scoring function based on applicants' normalized GPA (g) and SAT score (s), initially setting $f(c) = 0.5 \times g + 0.5 \times s$ and admitting the top-500 applicants. However, this may yield only 150 women whereas the fairness constraint is that at least 40% women should be admitted, possibly due to a gender disparity in the SAT score [44]. To meet the fairness constraint while preserving the explainability, the officer searches for a nearby fair scoring function with weights in the range of $[0.45, 0.55]$. An algorithm may find a fair function $f(c) = 0.55 \times g + 0.45 \times s$; if no such function exists, it reports failure, indicating the equal-weight design may be unsuitable. The officer may then adjust the design and, with the algorithm's help, obtain a fair function $f(c) = 0.6 \times g + 0.4 \times s$.*

Prior work has explored finding such fair scoring functions [6], but existing algorithms scale poorly with dataset size and worse with dimensionality. Since the general fairness constraints considered in [6] may introduce additional overhead, we focus on the simplest possible one on top-k selection, which ignores the relative rankings within the top-k (as in Example 1). For such a simple fairness constraint, do we have better algorithms that are efficient for large, high-dimensional datasets in practice, preferably running in $O(n \cdot \text{polylog}(n))$ time?

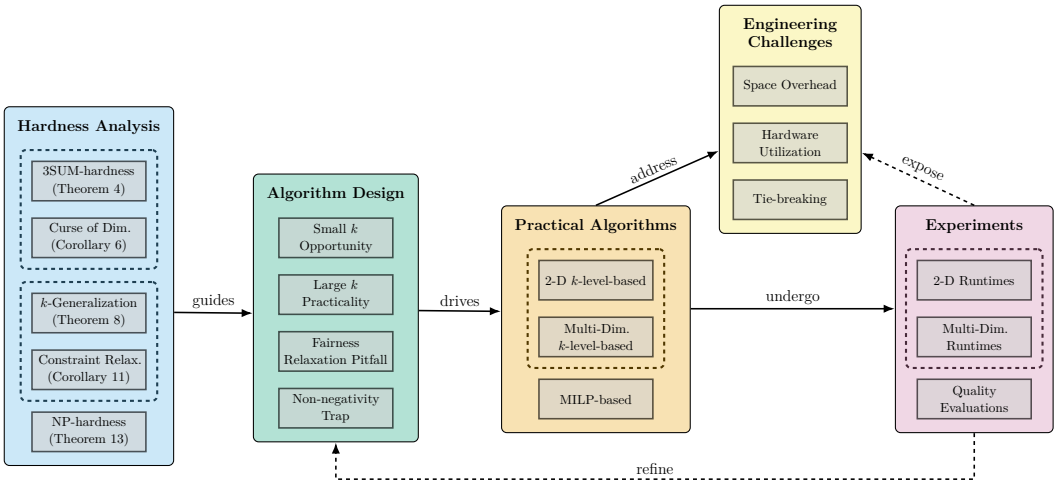


Figure 1 The structure and interplay of key results and components in this work. Solid arrows denote the primary workflow and direct influences, while dashed arrows indicate feedback.

Our contributions. To answer this question, we conduct an end-to-end, integrated study combining hardness analysis, algorithm design, practical engineering and empirical evaluation into a unified framework that yields an efficient solution. While some of the components may be of independent interest, their primary significance in this work lies in their interconnection. Our two-pronged solution, as well as many insights, emerge from the integration (see Figure 1).

Related work. Fair top-k selection under proportional fairness constraints has been widely studied [51, 31, 13, 6, 49] (see [52] for a review). Most existing work uses a single attribute or a fixed scoring function. Ref. [6] instead searched for a linear scoring function (i.e., a weight

vector) satisfying the fairness constraints while minimizing angular distance from a given one. Their method supports various fairness constraints but scales poorly with dataset size and dimensionality (see Section 4.2). Besides applying fairness constraints, ref. [35] introduced alpha-fairness – a metric capturing disparities between protected group proportions in the top-k subset and the entire dataset – and methods to find a weight vector optimizing this metric. However, their approach lacks a formal runtime analysis, still scales poorly, and breaks ties arbitrarily which may be unsuitable for fairness considerations (see Section 5.1).

2 Preliminaries

In the Fair Top-k Selection problem, we are given a set, $\mathcal{C}$, of candidates (or items), where each candidate $c \in \mathcal{C}$ has d scoring attributes and one sensitive attribute. The scoring attributes are represented by a point $p(c) = (p_1, p_2, \ldots, p_d) \in \mathbb{R}^d$, and the sensitive attribute by a label $A(c) \in \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m\}$. The label $\mathcal{G}_1$ denotes the protected group. Following [52], we also use $\mathcal{G}_i$ to denote the subset of candidates with $A(c) = \mathcal{G}_i$, particularly for $i = 1$.

We consider the class of linear scoring functions defined by a weight vector $w = (w_1, w_2, \ldots, w_d)$, where each $w_i \geq 0$ (a common assumption [46, 12, 6]). The score of a candidate c is computed as $f(p(c)) = w \cdot p(c) = \sum_{i=1}^{d} w_i p_i$. Since ties may occur, the top-k subset for a given w may not be unique. We consider all top-k subsets and regard w as fair if any of its top-k subsets satisfies the fairness constraint. This is embodied as follows:

▶ **Definition 2.** *For a given w and an integer $k \leq n$, a subset $\tau_k^w \subseteq C$ is a top-k subset if $|\tau_k^w| = k$ and for any pair of candidates that $c \in \tau_k^w$ and $c' \in C \setminus \tau_k^w$, $w \cdot p(c) \geq w \cdot p(c')$.*

We now formalize our notion of fairness. Let τ_k^w denotes a top-k subset for w. Let $L_k^{\mathcal{G}_1}$ (resp., $U_k^{\mathcal{G}_1}$) denote the desired lower (resp. upper) bound on the number of protected group candidates in τ_k^w, ideally near $k \cdot (|\mathcal{G}_1|/n)$. Then, w is a fair scoring weight vector if

$$L_k^{\mathcal{G}_1} \leq |\tau_k^w \cap \mathcal{G}_1| \leq U_k^{\mathcal{G}_1}. \tag{1}$$

Finally, one may wish to limit the search for a fair weight vector to a subset of the weight vector space. We denote this subset by V and describe it via a set of l linear inequalities. Without loss of generality, we further assume $||w||_1 = \sum_{i=1}^{d} w_i = 1$ (recall $w_i \geq 0$). Then, V can be effectively regarded as a subset of $\mathbb{R}^{d-1}$. Formally, we define our problem as follows:

▶ **Problem 3.** *We are given a set of n candidates with d scoring attributes and one sensitive attribute. Each candidate, c, has a point $p(c)$ in $\mathbb{R}^d$ and a label $A(c) \in \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m\}$. We are also given a non-negative integer $k \leq n$, non-negative integers $L_k^{\mathcal{G}_1}$ and $U_k^{\mathcal{G}_1}$, and a subset V of the weight vector space described by l linear inequalities. The goal is to find a real-valued weight vector $w = (w_1, w_2, \ldots, w_d) \in V$, where $w_i \geq 0$ for all i and $||w||_1 = 1$, such that*

$$L_k^{\mathcal{G}_1} \leq |\tau_k^w \cap \mathcal{G}_1| \leq U_k^{\mathcal{G}_1}.$$

Although our formulation allows $p(c)$ to be in $\mathbb{R}^d$, in practice, scoring attribute values are typically normalized to $[0, 1]^d$, as they are equivalent via uniform translation and scaling.

3 Hardness

Known algorithms for the Fair Top-k Selection problem [6] do not scale well with the dataset size, especially in higher dimensions. We establish (conditional) lower bounds that shed light on this limitation. In particular, we focus here more on the **practical implications** of these hardness results (Remarks within the section), rather than on their proofs ([11, Section 3]).

3.1 Hardness results in fixed dimensions

We show that for any fixed dimension, $d \geq 3$, the Fair Top-k Selection problem is 3SUM-Hard [47, 22], hence is unlikely to be solvable in $O(n^{2-\delta})$ time for any constant $\delta > 0$. Furthermore, based on a result in [20], we show that the problem is unlikely to be solvable in $O(n^{d-1})$ time for any $d \geq 3$. All our conditional lower bounds are established for a special case of the problem, where k is approximately $n/2$, $\mathcal{G}_1$ contains only one candidate, and $L_k^{\mathcal{G}_1} = U_k^{\mathcal{G}_1}$.

For the 3SUM-hardness, it is known that deciding whether three points in a given set in $\mathbb{R}^2$ lie on a line (a.k.a point collinearity problem) is 3SUM-Hard [22]. By a reduction from the collinearity problem to the Fair Top-k Selection problem, we show the following:

▶ **Theorem 4** (3SUM-hardness). *For any $d \geq 3$, Fair Top-k Selection cannot be solved in $O(n^{2-\delta})$ time for any constant $\delta > 0$, assuming the 3-SUM hypothesis.*

▶ **Remark 5.** *For large datasets, practical efficiency typically requires algorithms with run times $O(n \cdot \text{polylog}(n))$. However, our reduction suggests that such an efficient algorithm is unlikely to exist for general problem instances in dimensions $d \geq 3$.*

The point collinearity problem is a special case of the Affine Degeneracy problem in $\mathbb{R}^d$: deciding whether $d+1$ points lie on a hyperplane. Using a similar approach we have:

▶ **Corollary 6** (Curse of Dimensionality). *For any constant $d \geq 3$, Fair Top-k Selection in $\mathbb{R}^d$ is at least as hard as the Affine Degeneracy problem in $\mathbb{R}^{d-1}$.*

▶ **Remark 7.** *In a certain decision tree model, all algorithms for the Affine Degeneracy problem require $\Omega(n^d)$ time [20]. This suggests a lower bound of $\Omega(n^{d-1})$ for Fair Top-k Selection in $\mathbb{R}^d$ for $d \geq 3$. However, the reduction only applies if d is a constant, since the number of constructed instances also grows exponentially with d (see Section 3.3).*

3.2 Hardness results for more general cases

The conditional lower bounds in fixed dimensions are established using a special case of the problem. Here, we show that our conditional lower bounds also hold when the value of k, the number of candidates of $\mathcal{G}_1$, and the fairness constraint are more general and realistic.

We begin with definitions of key concepts. For a point set $S \subseteq \mathbb{R}^d$, a point $q \in \mathbb{R}^d$ is a *dominated* (resp. *dominating*) point of S if for any point $p \in S$, $q_i < p_i$ (resp. $q_i > p_i$) for all $1 \leq i \leq d$. For any non-negative weight vector, a dominated (resp. dominating) point has a lower (resp. higher) score than any point in S. We say a candidate c is a *dominated* (resp. *dominating*) candidate of a candidate set C if $p(c)$ is a dominated (resp. dominating) point.

With these concepts, we first consider a general k value. Suppose that we have an efficient algorithm that can only handle $k = \beta n$, where $0 < \beta < 1$ is a fixed constant. We show that it can also be applied to solve the instance in Corollary 6 efficiently.

▶ **Theorem 8** (k-Generalization). *Given a polynomial-time algorithm for Fair Top-k Selection with $k = \beta n$, where $0 < \beta < 1$ is a fixed constant, one can solve the Fair Top-k Selection with $k = (n-1)/2 - d + 1$ in the same asymptotic run time.*

Proof. Since k is expected to be small in practice, we consider the case $\beta n < (n-1)/2 - d + 1$ first. We construct C' from C by adding dominated candidates, making the size of C' to be $(n-1)/2\beta - (d-1)/\beta$. Then, we run the algorithm for Fair Top-k Selection with $k = \beta n$ on C'. Given the size of C', we have $k = (n-1)/2 - d + 1$. Since all candidates in $C' \setminus C$

are dominated ones, none of them will be in the top-k subset for any $k \leq n$. So, the top-k selection result will only contain candidates in C. The reduction takes $O(n)$ time with β being a constant. The case $\beta n > (n-1)/2 - d + 1$ can also be similarly handled. ◀

▶ **Remark 9.** *This reduction extends the lower bounds in Section 3.1 to a broad range of k where $k = \beta n$. However, for a sufficiently small k (e.g., $k = O(\text{polylog}(n))$), the reduction breaks down as C' requires adding too many candidates, causing it to take more than $O(n)$ time. This reveals that the previous lower bounds may not be applicable for a small k.*

Next, we consider a larger number of candidates of $\mathcal{G}_1$ and more general fairness constraints. Consider a dataset with two groups, where n_1 and n_2 denote the number of candidates of $\mathcal{G}_1$ and the other group. By adding both dominated and dominating candidates, we have:

▶ **Theorem 10** (Group-balancing). *Given a polynomial-time algorithm for Fair Top-k Selection with $n_1 = n_2 = n/2$ and $L_k^{\mathcal{G}_1} = U_k^{\mathcal{G}_1} = k \cdot (n_1/n) = k/2$, one can solve the Fair Top-k Selection with $n_1 = 1$ and $L_k^{\mathcal{G}_1} = U_k^{\mathcal{G}_1} = 1$ in the same asymptotic run time.*

Theorem 10 still has a restriction on the fairness constraint that $L_k^{\mathcal{G}_1} = U_k^{\mathcal{G}_1}$. By carefully manipulating the number of dominated and dominating candidates, we are able to show:

▶ **Corollary 11** (Constraint Relaxation). *Given a polynomial-time algorithm for Fair Top-k Selection with $L_k^{\mathcal{G}_1} = k \cdot (n_1/n) - \sigma_L$ and $U_k^{\mathcal{G}_1} = k \cdot (n_1/n) + \sigma_U$, where σ_L and σ_U are non-negative integers, one can solve the Fair Top-k Selection with $n_1 = 1$ and $L_k^{\mathcal{G}_1} = U_k^{\mathcal{G}_1} = 1$ in the same asymptotic run time.*

▶ **Remark 12.** *Intuitively, relaxing the fairness constraint should make the problem computationally easier. However, our reduction suggests that this is unlikely to be the case, as better algorithms imply a violation of the lower bounds established in Section 3.1.*

3.3 Hardness result in arbitrary dimensions

Corollary 6 suggests a possible extension to arbitrary dimensions, since Affine Degeneracy is NP-hard in arbitrary dimensions [20]. However, a direct extension fails because the reduction requires generating 2^{d-1} Fair Top-k Selection instances to circumvent the non-negativity of the weight vector. To resolve this issue, we use an alternative problem, Maximum Independent Set, for the reduction, starting from its binary integer programming formulation [36].

▶ **Theorem 13** (NP-hardness). *Fair Top-k Selection is NP-hard in arbitrary dimensions.*

▶ **Remark 14.** *The failure to extend Corollary 6 under non-negativity may appear to reveal a gap in the hardness barrier. Moreover, this non-negativity appears to permit efficient mathematical optimization techniques (Section 5.2), which typically operate outside the decision-tree model (Corollary 6). However, our NP-hardness result shows that such methods may not work efficiently for the worst case. Notably, the relaxation technique, essential to many efficient solvers for our formulation (Section 5.2), also plays a key role in the reduction.*

4 From hardness results to algorithm design

As stated, our two-pronged solution is guided by the hardness results. Here, we examine these insights in details. Also, we examine the case of small k that is critical to our solution.

4.1 Implications for algorithm design

Our hardness results, starting from restricted cases, show that the problem becomes hard to solve for $d \geq 3$ (Remark 5) and demonstrate the curse of dimensionality (Remark 7). Extending these results to more general cases yields key insights for algorithm design:

- **Small k Opportunity.** The reduction technique used in Theorem 8 breaks down for a sufficiently small k (Remark 9). This suggests the potential of a more efficient algorithm by explicitly leveraging k in the design.
- **Large k Practicality.** The hardness of the problem is robust for a large k, making the existence of worst-case efficient algorithms unlikely. Therefore, designing efficient algorithms for large k should focus on real-world performance.
- **Fairness Relaxation Pitfall.** Relaxing the fairness constraint may appear to make the problem easier, making an algorithm that starts with a loose constraint and gradually tightens it seems attractive. However, such relaxation does not reduce worst-case computational hardness (Remark 12), limiting its value.
- **Non-negativity Trap.** With the failure to extend Corollary 6, non-negativity of variables (weights) and most coefficients (scoring attribute values) appears to open the door to efficient mathematical programming techniques (Remark 14). However, our NP-hardness result shows that this non-negativity may not yield runtime improvements.

Drawing on these insights, we arrive at a two-pronged solution: a theoretically grounded algorithm for a small k, and a practically efficient algorithm for a large k. Next, we demonstrate that the lower bounds are indeed breakable for a small k.

4.2 The case of small k

The Fair Top-k Selection problem can be solved by the algorithms in [6], which run in $O(n^2(k + \log n))$ (improvable to $O(n^2 \log n)$) for $d = 2$ and $O(n^{2d+2})$ for a constant $d \geq 3$. In contrast, our algorithm explicitly incorporates k, allowing it to run fast when k is small.

Like [6], our algorithm employs a dual transformation, but we adopt a different one given in [12, 48] that is better suited to our setting. Let $p = (p_1, p_2, \ldots, p_d)$ be $p(c)$ of a candidate $c \in C$. We transform p into a hyperplane $H(p)$ defined by ($x_1, x_2, \cdots x_d$ are variables)

$$(p_1 - p_d)x_1 + (p_2 - p_d)x_2 + \cdots + (p_{d-1} - p_d)x_{d-1} - x_d + p_d = 0. \tag{2}$$

Given the dual transformation, *k-level* and *cell* can be defined accordingly ([11, Section 4.2]). One important property of the k-level is that if a hyperplane has a cell that is part of the $(k-1)$-level, then the primal point corresponding to the hyperplane is the kth point for some weight vectors, since there are exactly $k-1$ hyperplanes lying "above" in the x_d dimension.

We now proceed to the algorithm. Assume the $(k-1)$-level is constructed and cells adjacencies are identified, with hyperplanes in general position (i.e, no more than d hyperplanes intersect at a point), and ties in top-k selection broken arbitrarily. Our algorithm first finds an arbitrary weight vector $t \in V$ and locates a $(k-1)$-level cell intersected by the downward ray induced by t. Next, we obtain the corresponding top-k subset and count the number of candidates of $\mathcal{G}_1$ in the top-k subset, $n_k^{\mathcal{G}_1}$, and test it against $L_k^{\mathcal{G}_1}$ and $U_k^{\mathcal{G}_1}$. If the fairness constraint is satisfied, we output t. If not, we traverse the $(k-1)$-level by visiting adjacent cells. For each newly visited cell, we test if its projection onto the first $d-1$ coordinates intersects V. If it is, we update the $n_k^{\mathcal{G}_1}$ as the top-k subsets of adjacent cells differ in at most one candidate. We maintain the $n_k^{\mathcal{G}_1}$ during the traversal and whenever the top-k subset changes, test $n_k^{\mathcal{G}_1}$ against $L_k^{\mathcal{G}_1}$ and $U_k^{\mathcal{G}_1}$. The algorithm stops when either a fair weight vector is obtained from a $(k-1)$-level cell or all relevant $(k-1)$-level cells have been visited.

For the runtime analysis, leveraging known results of the k-level [37, 1, 24, 15], along with an application of an efficient convex polyhedra intersection algorithm [7], we have:

▶ **Theorem 15.** *In fixed dimensions, Fair Top-k Selection can be solved in $O(nk^{1/3}\log n)$ time for $d = 2$ and in $O(nk^{3/2}\log^4 n)$ time for $d = 3$, both with $l = O(n)$. In $d \geq 4$ dimensions, it can be solved in $O(n^{\lfloor d/2\rfloor}k^{\lceil d/2\rceil})$ (expected) time with a sufficiently small l.*

▶ **Remark 16.** *Notice that the algorithm relies on assumptions that do not fully align with the problem setting (Section 5.1). With careful refinements and thorough analysis, one can show that the bounds still hold. However, as the algorithm chiefly illustrates the key component for breaking the lower bounds, its full resolution is deferred to the practical context (Section 5.1).*

5 Practical algorithms

The k-level-based algorithm above builds on theoretically efficient algorithms for constructing the k-level, yet few have been empirically evaluated. Moreover, existing studies of a related algorithm show worse empirical performance than simpler heuristics [2, 25]. To make it practical, we identify key engineering challenges and present corresponding solutions. For large k, our algorithm design principle shifts from theoretical guarantees to practical efficiency. As a result, we present an algorithm based on mixed-integer linear programming (MILP).

5.1 Practical k-level-based algorithms

We identify three key engineering challenges for k-level-based algorithms to become practical:

- **Space Overhead.** The k-level-based algorithm constructs the entire k-level, whose structural complexity may grow quickly with n and d. For large datasets, it may exhaust internal memory, requiring much slower external-memory solutions. Yet users may seek a fair weight vector within a small, explainable region, making a full k-level unnecessary.
- **Hardware Utilization.** A prior theoretically efficient algorithm [25] exhibits bad memory performance, which contributes to its slow run time in practice. Moreover, many algorithms are not naturally parallelizable and cannot exploit modern multi-core systems. Bridging the theory-practice gap for parallel algorithms is even more challenging [9].
- **Tie-breaking.** Most theoretical algorithms for constructing the k-level assume general position (limiting ties among candidates) and break ties arbitrarily for simplicity. However, for the fair top-k selection, tie-breaking can affect the number of protected group candidates selected, impacting fairness. In our problem setting, an algorithm must be designed to account for all possible tie-breaking outcomes when evaluating fairness.

Next, we present practical k-level-based algorithms, focusing on our methods addressing these challenges. (Tie-breaking will be revisited in Section 5.2 for the MILP-based algorithm.)

5.1.1 2-D k-level-based algorithm

Our algorithm in 2-D is an adaptation of an algorithm in [14] (briefly outlined in [8]). The algorithm is a sweep-line algorithm using two kinetic priority queues, S_1 and S_2, to keep track of the k-level cell. Viewing the x-coordinate of the sweep-line as "time", S_1 (resp. S_2) contains the k (resp. $n-k$) lines lying above (resp. below) at the time instant. The algorithm proceeds by processing internal queue events or updating queues during the sweep.

Space overhead. We note that in this algorithm, S_1 actually maintains the top-k subset during the sweep. Instead of explicitly constructing the $(k-1)$-level, one can simply check whether the top-k subset induced by S_1 satisfies the fairness constraint. Even better, following the theoretical algorithm (Section 4.2), one can use a variable to keep track of the number of lines of $\mathcal{G}_1$ in S_1. The space usage reduces to $O(n)$ by avoiding the explicit construction.

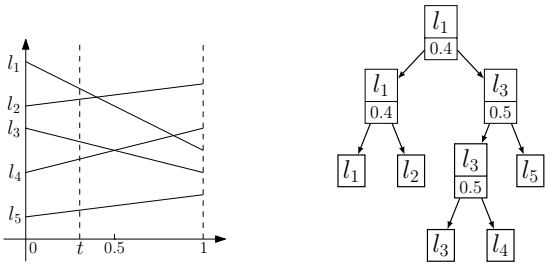


Figure 2 Maximum kinetic tournament tree (right) for lines (left) at $x = t$. We augment the tree with the replacement operation, which can be implemented by substituting the target line (i.e., l_1 in this example) at the leaf level and propagating updates along the path to the root.

Hardware utilization. Our low-level optimizations focus on the kinetic priority queue, a critical component for the real-world performance (see discussions in Section 6.2.1). In [14], fully dynamic kinetic priority queues are used, which may exhibit bad memory performance. However, observe that the sizes of S_1 and S_2 are fixed throughout the sweep; to correctly update them, one only has to replace the minimum (resp. maximum) element of S_1 (resp. S_2) with the maximum (resp. minimum) element of S_2 (resp. S_1) at the "time" instant of crossing. We use the kinetic tournament tree [8] for implementing the kinetic priority queue, in which this replacement can be done in $O(\log n)$ time (see Figure 2). Since the tree configuration is fixed, we further flatten trees into linear arrays for better cache performance, using a pre-order traversal layout. (Other layouts [29] might work better but are not explored.)

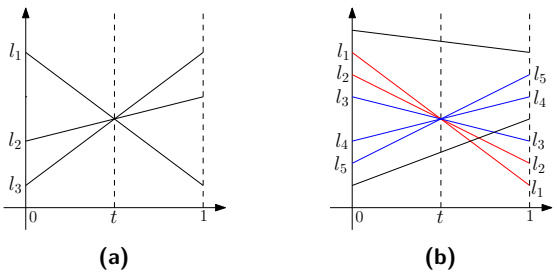


Figure 3 **(a)** Three lines intersect at $x = t$. The tie is broken by perturbing x to $x = t + \epsilon$. This, in fact, sorts the lines by their slopes, and orders l_3 ahead of l_1 and l_2 for a max-queue. **(b)** For $k = 3$, we have $l_1, l_2 \in S_1$ (red) and $l_3, l_4, l_5 \in S_2$ (blue) before the intersection. For correct configurations of S_1 and S_2 at $x = t + \epsilon$, one can exchange l_1 for l_5 and l_2 for l_4 between S_1 and S_2.

Tie-breaking. The algorithm in [14] assumes no more than two lines intersect at a point. To resolve this issue, we apply the symbolic perturbation [19], which provides a tie-breaking rule that compares the slopes of lines (see Figure 3a). Also, pairs of lines to exchange are carefully identified for correctly updating S_1 and S_2 (see Figure 3b). However, the top-k subset induced by the resulting S_1 is only one of the tie-breaking outcomes. To correctly

account for all tie-breaking outcomes, we collect all intersecting lines at this "time" instant (including duplicates) by a guided tree traversal. Each intersecting line can be found with an additional $O(\log n)$ cost. To avoid enumerating all possible top-k subsets, we compute tight lower and upper bounds on the number of $\mathcal{G}_1$ members and use these bounds to test fairness.

5.1.2 Multi-dimensional k-level-based algorithm

Our multi-dimensional algorithm adapts the method in [4], later implemented in [5] for rank-regret minimization. Although not as fast as the best theoretical algorithm, it may be better at exploiting modern multi-core systems as suggested in [4]. However, no parallel implementation currently exists. Starting from a cell and its top-k subset, the algorithm explores adjacent cells by testing potential top-k subsets, formed by swapping one top-k element with an outsider. The test can be done by finding a separating hyperplane via a linear program. This enables enumeration of all valid top-k subsets in a breadth-first manner.

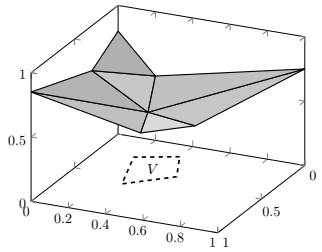


Figure 4 $(k-1)$-level (triangular mesh) and V (dashed region on the x-y plane) in 3-D. Testing cell intersection is equivalent to finding a point w in V whose downward-directed ray from $(w_1, w_2, +\infty)$ hits the cell (triangle). The constraints (points in V) can be incorporated into the linear program.

Space overhead. We incorporate weight vector space constraints (i.e., V) into each linear program, restricting it to find only weight vectors within V when testing a potential top-k subset (see Figure 4). This approach reduces the number of cells (i.e., top-k subsets) visited during the traversal, and thus space usage, especially when V only defines a small region.

Hardware utilization. Notice that the algorithm can be regarded as a breadth-first search algorithm on a graph, where each cell is a node and each edge connects a pair of adjacent cells. Therefore, a method for parallel BFS can be applied. However, since the graph is not known explicitly and finding an adjacent node can be expensive, efficient but sophisticated level-synchronous parallel algorithms (e.g. [34, 41]) may be difficult and less beneficial to adapt. Here we take a simpler approach by distributing jobs for finding adjacent nodes. Each thread independently takes a job for testing whether the given potential top-k subset is valid. If it is, it generates subsequent jobs. We use a global job queue for parallel job distribution. To reduce synchronization costs, a lock-free queue with relaxed FIFO ordering [30] (and a memory reclamation scheme [10]) is used, since a strict FIFO order is not needed. To keep track of the processed top-k subsets, we use a lock-free hash table [40]. Since the table size increases monotonically, we use an insertion-only table with memory reclamation disabled for further reducing runtime overhead. Notably, our lockless approach improves efficiency but introduces subtle implementation issues. For example, preventing premature thread termination requires carefully accounting for dequeue and emptiness-check linearization points [27] (specifically, under the restricted out-of-order specification [26]) of the job queue.

Tie-breaking. We modify linear programs to allow points lying on the hyperplane rather than enforcing strict separation. This approach enables the algorithm to enumerate all possible top-k subsets, as a top-k subset with ties (typically) corresponds to a lower-dimensional cell (see [11, Section 5.1.2]). In the extreme case, the run time will increase by a lot, but this rarely occurs in practice. Nevertheless, we still trade some efficiency for ease of implementation.

5.2 Mixed-integer linear programming-based algorithm

For a large k value, our focus becomes designing a practically efficient algorithm, which results in an algorithm based on MILP. We note that MILP is NP-hard and all known algorithms run in exponential times in the worst case. However, in practice, an MILP solver can be quite fast, and it was shown that an MILP-based algorithm is efficient for finding a linear scoring function for the reverse top-k query [16], making it an approach worth considering.

For simplicity, assume without loss of generality that each scoring attribute value is in the range of $[0, 1]$. We define a indicator variable $\delta_c \in \{0, 1\}$ for each candidate c satisfying

$$-1 \leq w \cdot p(c) - \lambda - \delta_c \leq 0 \tag{3}$$

for any given weight vector w and a cut-off value $\lambda \in [0, 1]$. δ_c encodes the top-k subset membership of c for a (w, λ) pair defining a separating hyperplane (see [11, Section 5.2]).

To ensure a value of λ will be infeasible unless it yields k candidates with $\delta_c = 1$, the constraint $\sum_c \delta_c = k$ is applied. Moreover, the fairness constraint can be encoded as follows:

$$L_k^{\mathcal{G}_1} \leq \sum_{A(c)=\mathcal{G}_1} \delta_c \leq U_k^{\mathcal{G}_1}. \tag{4}$$

Finally, constraints on the weight vector (i.e., V) can also be easily incorporated. We arrive at a MILP with a weight vector, a cut-off value, and n indicator variables as its unknown variables. Finding a fair weight vector becomes finding a feasible solution to the MILP.

Now, let us revisit the "Non-negativity Trap" (see Section 4.1). In our MILP formulation, all variables (w, λ and δ_c) and most of the coefficients ($p(c)$) are non-negative. The main issue is the negative coefficient of λ, which prevent direct adaptations of the techniques such as those in [3, 43]. While one might hope to circumvent this inconvenience, our NP-hardness result show that the problem may be fundamentally intractable in the worst case. This also suggests that the practical success of the MILP-based algorithm is largely heuristic.

For our implementation, we use Gurobi [23] – a state-of-the-art, mathematical programming tool – to solve the MILP. This solver also exploits parallelism to enhance performance.

Tie-breaking. The tie-breaking consideration has already been embodied in our MILP formulation which naturally addresses it. Notice that if $w \cdot p(c) = \lambda$, δ_c is free to choose its value. If the top-k subset is not unique for a weight vector, all possible top-k subsets will be considered in the MILP. The fairness constraint is applied to further restrict values of δ_c.

6 Experiments

We present experimental results on real-world datasets, including both run times and quality evaluations. We conclude with key observations and discussions of algorithm selection.

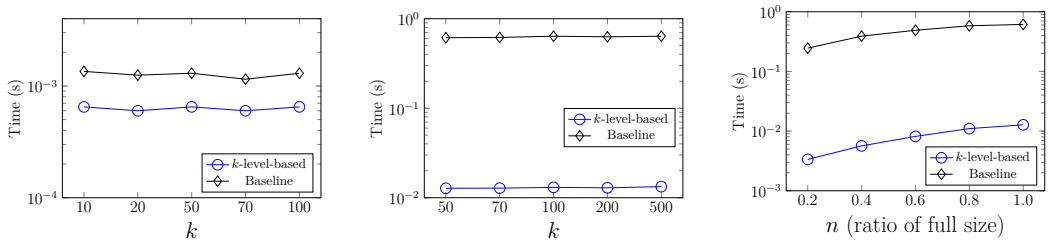
6.1 Experimental setup

In our experiments, a weight vector w^o is provided as an input. If w^o is unfair, the algorithm is run to find a "nearby" fair weight vector w^f. More formally, for a given unfair w^o, one aims to find a fair w^f such that $|w_i^f - w_i^o| \leq \epsilon$ for all $1 \leq i \leq d$, where ϵ is user-specified parameter that limits the allowable deviation. A failure is reported if no such fair w^f exists. We used two real-world datasets, *COMPAS* ($n = 7214$, $2 \leq d \leq 6$) [38] and *IIT-JEE* ($n = 384\,977$, $2 \leq d \leq 3$) [28], with preprocessing as needed. 2DRAYSWEEP and ATC$_+$ from [6] were implemented as baselines, with several enhancements incorporated. By default, $k = 50$, with $\epsilon = 0.1$ for $d = 2$ and $\epsilon = 0.05$ for $d \geq 3$. See [11, Section 6.1] for the full experimental setup and `https://github.com/caiguangya/fair-topk` for the code, data, and container [33].

6.2 Runtime experiments

Since our main focus is execution speed, we start with runtime experiments and highlight speedups from better hardware utilization that also enable more informed comparisons [32].

6.2.1 Runtime experiments for 2-D datasets



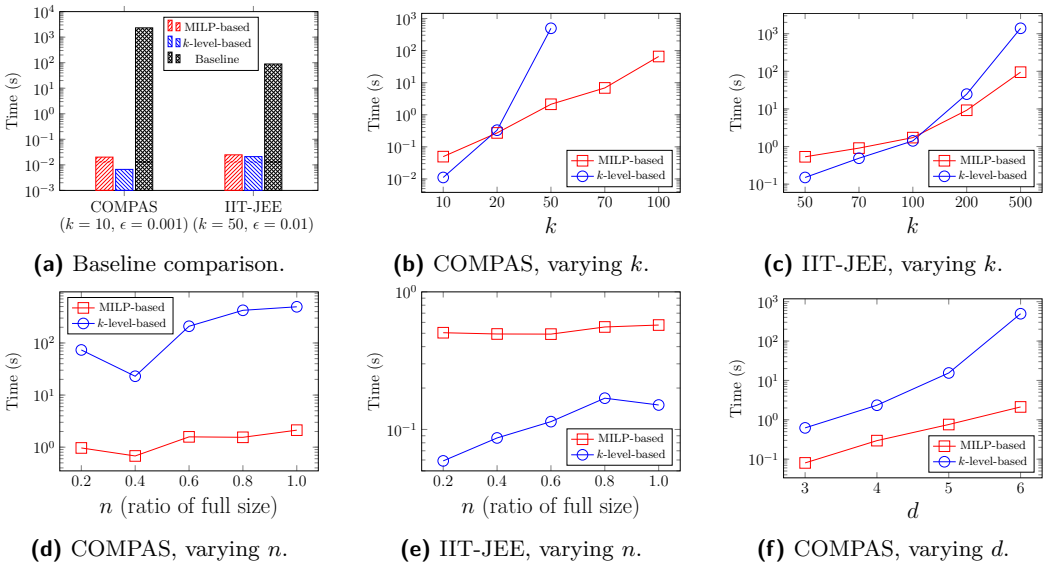
(a) COMPAS, varying k. **(b)** IIT-JEE, varying k. **(c)** IIT-JEE, varying n.

Figure 5 Runtime experimental results for 2-D datasets.

Varying k. As shown in Figures 5a and 5b, our k-level-based algorithm consistently outperformed the baseline, achieving over 40x speedups on the larger dataset (IIT-JEE). Notably, its performance remained stable as k increased. In fact, the line-sweeping step only contributed a small fraction of the total run time in our experiments. The algorithm [14] we adapted is output-sensitive, running efficiently when the actual k-level complexity is low. Moreover, it is known that the best known complexity of the k-level in 2-D, i.e., $O(nk^{1/3})$ [18], might not be a tight one [45], and the "average" complexity can be significantly lower [17].

Varying n. Figure 5c shows run times with n controlled by sampling 20%–100% of the IIT-JEE dataset. For the baseline, the run time increased mildly as n increased, performing better than its worst-case time complexity would suggest. However, our algorithm was still substantially faster. Comparing two algorithms side-by-side, both exchange orders of lines during the sweep, but the baseline considers all pairs, while our k-level-based approach uses kinetic tournament trees to skip many exchanges, directly reducing their number. Since many tree operations incur an $O(\log n)$ overhead, efficient implementation becomes critical. The experiments reveal an area where an efficient implementation should focus, and our optimizations (Section 5.1.1) addresses this need effectively (Section 6.2.3), enhancing the algorithm's performance in scenarios where the worst-case behavior does not manifest.

6.2.2 Runtime experiments for multi-dimensional datasets



(a) Baseline comparison. **(b)** COMPAS, varying k. **(c)** IIT-JEE, varying k.

(d) COMPAS, varying n. **(e)** IIT-JEE, varying n. **(f)** COMPAS, varying d.

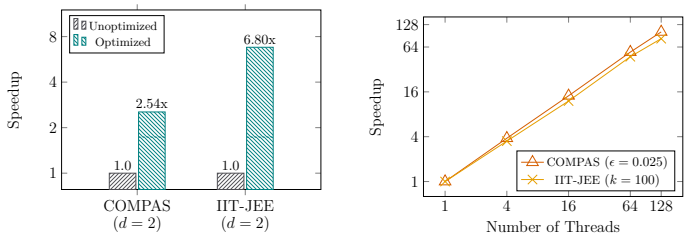
Figure 6 Runtime experimental results for multi-dimensional datasets ($3 \leq d \leq 6$).

Baseline comparison. In higher-dimensional experiments, the baseline regularly failed to finish within the time limit, so a smaller value of ϵ was used. As shown in Figure 6a, for the IIT-JEE dataset ($d = 3$), our MILP-based and k-level-based algorithms achieved 3598x and 4222x speedups, respectively. For the COMPAS dataset ($d = 6$), we had to further lower the value of ϵ, and our algorithms were five orders of magnitude faster. Due to the inefficiency of the baseline, subsequent experiments report only the run times of our two algorithms.

Varying k. As shown in Figures 6b and 6c, the MILP-based algorithm outperformed the k-level-based one for larger k, especially on the higher-dimensional dataset (COMPAS), while the k-level-based algorithm was faster for smaller k and lower dimensions (IIT-JEE). These results align with known bounds on the k-level complexity for $d \geq 3$. Unlike in 2-D, the multi-dimensional k-level algorithm's run time is more dependent on k, as each top-k subset will generate $O(nk)$ linear programs. Notably, the MILP-based algorithm also performed well on the lower-dimensional dataset (IIT-JEE) despite a larger number of (indicator) variables.

Varying n. As shown in Figures 6d and 6e, the run time of the k-level-based algorithm generally increased as n increased. However, some anomalies appeared since a larger n does not always yield more k-level cells, especially when only part of the k-level is explored. The MILP-based algorithm showed more stable run times despite increasing variable count.

Varying d. Figure 6f shows run times with d controlled by selecting subsets of COMPAS scoring attributes after preprocessing. The run times of both algorithms increased significantly as d increased. For the k-level-based algorithm, this aligns with the known results of k-level structural complexity. Besides, the $O(d!n)$ cost of the linear programming algorithm [39] we used also contributed to the run time increase. The MILP-based algorithm also showed significant slowdown, despite the number of variables only increasing by 1 as d increased by 1.



(a) Memory access. **(b)** Parallelism.

Figure 7 Performance gains from better hardware utilization, measured by total run time.

Notably, for $d = 3$, the k-level-based algorithm was slower than the MILP-based algorithm – unlike what we observed on the IIT-JEE dataset with the default $k = 50$ (Figure 6c). We note that this 3-D dataset created by selecting three attributes contains many candidates with identical scoring attribute values, which exposed the tie-breaking inefficiency in our implementation. However, since this lower-dimensional dataset is derived from a higher-dimensional one, it may contain fewer scoring attributes than intended to distinguish each candidate. Therefore, the slowdown is more likely an artifact rather than a real-world issue.

6.2.3 Hardware utilization

Memory access. Figure 7a shows the performance gains from optimizing the tournament tree of the 2-D k-level-based algorithm. We compared optimized and unoptimized versions, with the latter approximating a fully dynamic tournament tree while avoiding self-balancing by inserting new nodes into vacancies left by deletions. Our experimental results show notable speedups of 2.54x and 6.80x respectively, highlighting the benefits of our optimizations.

Parallelism. Figure 7b shows the efficiency of our parallel implementation of the multi-dimensional k-level-based algorithm. k and ϵ were default unless specified. Our implementation showed strong parallel scalability up to 128 threads, achieving speedups of 102x and 83x over the (stand-alone) sequential implementation on COMPAS and IIT-JEE, respectively.

6.3 Quality evaluation experiments

The quality was evaluated using the following metrics: (1) **Found/Unfair ratio**: the ratio between the number of input unfair weight vectors and found fair ones. (2) $\boldsymbol{w}$ **difference**: the average L_1 distance between w^o and w^f. (3) $\boldsymbol{\mathcal{G}_1}$ **proportion**: the average $\mathcal{G}_1$ candidate proportion in the top-k subset derived from w^f. (4) **Utility loss**: the average relative utility difference between the top-k subset derived from w^o and w^f. (See [11, Section 6.3].)

Table 1 shows the experimental results. We note that the k-level-based algorithm achieved better results for the w difference and utility loss. In fact, these two metrics are related, as a closer w^f to w^o typically yields a top-k subset with high-utility candidates. Our k-level-based algorithm naturally favors "nearby" weight vectors due to its breadth-first search approach with w^o as the starting point, while the MILP-based algorithm does not directly leverage w^o.

6.4 Observation summary and algorithm selection

We summarize key experimental observations and discuss algorithm selection based on them.

Table 1 Outcome quality evaluation for COMPAS dataset ($k = 50$).

ϵ	Fairness constraint	Found/Unfair ratio	w difference		$\mathcal{G}_1$ proportion		Utility loss	
			k-level	MILP	k-level	MILP	k-level	MILP
0.05	[34%, 66%]	19/39	**0.149**	0.211	66.0%	**65.7%**	**0.476%**	1.210%
	[40%, 60%]	8/44	**0.144**	0.213	59.8%	**59.5%**	**0.449%**	1.328%
	[44%, 56%]	6/47	**0.155**	0.202	**55.3%**	55.6%	**0.718%**	1.319%
0.025	[40%, 60%]	3/44	**0.088**	0.090	60.0%	**59.3%**	**0.121%**	0.142%
0.05		8/44	**0.144**	0.213	59.8%	**59.5%**	**0.449%**	1.328%
0.075		18/44	**0.247**	0.308	59.9%	**59.7%**	**2.416%**	3.187%

2-D. The k-level-based algorithm is the preferred method, with following key observations:

- Run times remained stable across k for both datasets, suggesting that k-level structural complexity may have limited impact on run time in real-world 2-D scenarios.
- Varying n experiments indicate that the k-level-based algorithm's advantage over the baseline may stem from the use of the kinetic priority queue that directly reduces the number of pairwise line exchanges, rather than the worst-case run time improvement.

In particular, building on the second observation, our implementation of the k-level-based algorithm was purposely engineered to further improve the performance (Figure 7a).

Multi-dimensions. By our algorithm design principle (Section 4), determining a threshold value of k is crucial. However, some previous discussions may not apply here, as our concern shifts to real-world performance. For real-world datasets, we have following key observations:

- The k-level-based algorithm outperformed the MILP-based algorithm for a small k, but its run time grew more rapidly with k, unlike its stable performance in 2-D.
- Run times of both algorithms increased dramatically with dimensionality d, with the k-level-algorithm increasing more rapidly, partly due to the LP algorithm used.
- The k-level-based algorithm performed better on the IIT-JEE dataset but worse on the 3-D COMPAS dataset with $k = 50$ (Figures 6c and 6f), likely due to an artifact exposing an inefficiency in our implementation for tie-breaking.

For algorithm selection, the first two observations suggest that as d increases, the threshold value of k at which the MILP-based algorithm becomes superior is likely to decrease. However, the last one suggests that performance comparisons between the two can be influenced by the underlying data distribution. Precisely defining a threshold value of k remains non-trivial, as the performance of the MILP-based algorithm is inherently heuristic. Nonetheless, our experimental results still indicate that this threshold is inversely correlated with d.

7 Conclusion

We presented an integrated study on finding a fair linear scoring function for top-k selection. Our hardness analysis established the theoretical foundation and guided the algorithm design, leading to our two-pronged solution: A k-level-based algorithm for small k and a MILP-based algorithm for large k. Practical engineering then addressed implementation challenges to ensure the solution works in practice. Finally, empirical evaluation explored real-world scenarios, identifying areas critical to practical performance. (See Figure 1 for the roadmap.) Our result is a practically efficient solution that is significantly faster than the state of the art, accompanied by many insights which we believe can inform future studies.

References

1. Pankaj K. Agarwal and Jiří Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13(4):325–345, 1995. doi:10.1007/BF01293483.
2. Yuval Aharoni, Dan Halperin, Iddo Hanniel, Sariel Har-Peled, and Chaim Linhart. On-line zone construction in arrangements of lines in the plane. In *Proceedings of the 3rd International Workshop on Algorithm Engineering (WAE)*, pages 139–153, 1999. doi:10.1007/3-540-48318-7_13.
3. Shabbir Ahmed and Weijun Xie. Relaxations and approximations of chance constraints under finite distributions. *Mathematical Programming*, 170:43–65, 2018. doi:10.1007/S10107-018-1295-Z.
4. Artur Andrzejak and Komei Fukuda. Optimization over k-set polytopes and efficient k-set enumeration. In *Proceedings of the 6th International Workshop on Algorithms and Data Structures (WADS)*, pages 1–12, 1999. doi:10.1007/3-540-48447-7_1.
5. Abolfazl Asudeh, Gautam Das, H. V. Jagadish, Shangqi Lu, Azade Nazi, Yufei Tao, Nan Zhang, and Jianwen Zhao. On finding rank regret representatives. *ACM Transactions on Database Systems*, 47(3):1–37, 2022. doi:10.1145/3531054.
6. Abolfazl Asudeh, H. V. Jagadish, Julia Stoyanovich, and Gautam Das. Designing fair ranking schemes. In *Proceedings of the 2019 international conference on management of data (SIGMOD)*, pages 1259–1276, 2019. doi:10.1145/3299869.3300079.
7. Luis Barba and Stefan Langerman. Optimal detection of intersections between convex polyhedra. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1641–1654, 2014.
8. Julien Basch, Leonidas J. Guibas, and G. D. Ramkumar. Reporting red-blue intersections between two sets of connected line segments. In *Proceedings of The Fourth Annual European Symposium on Algorithms (ESA)*, pages 302–319, 1996. doi:10.1007/3-540-61680-2_64.
9. Guy Blelloch, William Dally, Margaret Martonosi, Uzi Vishkin, and Katherine Yelick. SPAA'21 panel paper: Architecture-friendly algorithms versus algorithm-friendly architectures. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 1–7, 2021. doi:10.1145/3409964.3461780.
10. Trevor Alexander Brown. Reclaiming memory for lock-free data structures: There has to be a better way. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 261–270, 2015. doi:10.1145/2767386.2767436.
11. Guangya Cai. Finding a fair scoring function for top-k selection: From hardness to practice, 2026. arXiv:2503.11575.
12. Wei Cao, Jian Li, Haitao Wang, Kangning Wang, Ruosong Wang, Raymond Chi-Wing Wong, and Wei Zhan. k-regret minimizing set: Efficient algorithms and hardness. In *20th International Conference on Database Theory (ICDT)*, pages 11:1–11:19, 2017. doi:10.4230/LIPIcs.ICDT.2017.11.
13. L Elisa Celis, Damian Straszak, and Nisheeth K Vishnoi. Ranking with fairness constraints. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 28:1–28:15, 2018. doi:10.4230/LIPIcs.ICALP.2018.28.
14. Timothy M. Chan. Remarks on k-level algorithms in the plane, 1999. Manuscript. Available at https://tmc.web.engr.illinois.edu/lev2d_7_7_99.pdf.
15. Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. *Discrete & Computational Geometry*, 64(4):1235–1252, 2020. doi:10.1007/S00454-020-00229-5.
16. Zixuan Chen, Panagiotis Manolios, and Mirek Riedewald. Why not yet: Fixing a top-k ranking that is not fair to individuals. *Proceedings of the VLDB Endowment (VLDB)*, 16(9):2377–2390, 2023. doi:10.14778/3598581.3598606.
17. Man-Kwun Chiu, Stefan Felsner, Manfred Scheucher, Patrick Schnider, Raphael Steiner, and Pavel Valtr. On the average complexity of the k-level. *Journal of Computational Geometry*, 11(1):493–506, 2020. doi:10.20382/JOCG.V11I1A19.

18 Tamal K. Dey. Improved bounds for planar k-sets and related problems. *Discrete & Computational Geometry*, 19:373–382, 1998. doi:10.1007/PL00009354.

19 Herbert Edelsbrunner and Ernst Peter Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990. doi:10.1145/77635.77639.

20 Jeff Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM Journal on Computing*, 28(4):1198–1214, 1999. doi:10.1137/S0097539797315410.

21 Sorelle A. Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. The (im)possibility of fairness: Different value systems require different mechanisms for fair decision making. *Communications of the ACM*, 64(4):136–143, 2021. doi:10.1145/3433949.

22 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995. doi:10.1016/0925-7721(95)00022-2.

23 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL: https://www.gurobi.com.

24 Dan Halperin and Micha Sharir. Arrangements. In *Handbook of discrete and computational geometry*, pages 723–762. Chapman and Hall/CRC, 2017.

25 Sariel Har-Peled. Taking a walk in a planar arrangement. *SIAM Journal on Computing*, 30(4):1341–1367, 2000. doi:10.1137/S0097539799362627.

26 Thomas A. Henzinger, Christoph M. Kirsch, Hannes Payer, Ali Sezgin, and Ana Sokolova. Quantitative relaxation of concurrent data structures. In *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL)*, pages 317–328, 2013. doi:10.1145/2429069.2429109.

27 Maurice P. Herlihy and Jeannette M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 12(3):463–492, 1990. doi:10.1145/78969.78972.

28 Indian Institute Of Technology. IIT-JEE, 2009. Retrieved from https://indiankanoon.org/doc/1955304.

29 Paul-Virak Khuong and Pat Morin. Array layouts for comparison-based searching. *ACM Journal of Experimental Algorithmics*, 22:1–39, 2017. doi:10.1145/3053370.

30 Christoph M. Kirsch, Michael Lippautz, and Hannes Payer. Fast and scalable, lock-free k-fifo queues. In *Proceedings of the 12th International Conference on Parallel Computing Technologies (PaCT)*, pages 208–223, 2013. doi:10.1007/978-3-642-39958-9_18.

31 Jon Kleinberg and Manish Raghavan. Selection problems in the presence of implicit bias. In *9th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 33:1–33:17, 2018. doi:10.4230/LIPIcs.ITCS.2018.33.

32 Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowledge and Information Systems*, 52(2):341–378, 2017. doi:10.1007/S10115-016-1004-2.

33 Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. Singularity: Scientific containers for mobility of compute. *PloS one*, 12(5):e0177459, 2017.

34 Charles E. Leiserson and Tao B. Schardl. A work-efficient parallel breadth-first search algorithm (or how to cope with the nondeterminism of reducers). In *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures (SPAA)*, pages 303–314, 2010. doi:10.1145/1810479.1810534.

35 Hao Liu, Raymond Chi-Wing Wong, Zheng Zhang, Min Xie, and Bo Tang. Fair top-k query on alpha-fairness. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 2338–2350, 2024. doi:10.1109/ICDE60146.2024.00185.

36 László Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979. doi:10.1109/TIT.1979.1055985.

37 Ketan Mulmuley. On levels in arrangements and voronoi diagrams. *Discrete & Computational Geometry*, 6:307–338, 1991. doi:10.1007/BF02574692.

38 ProPublica. Correctional Offender Management Profiling for Alternative Sanctions, 2014. Retrieved from `https://github.com/propublica/compas-analysis`.

39 Raimund Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry*, 6:423–434, 1991. `doi:10.1007/BF02574699`.

40 Ori Shalev and Nir Shavit. Split-ordered lists: Lock-free extensible hash tables. *Journal of the ACM (JACM)*, 53(3):379–405, 2006. `doi:10.1145/1147954.1147958`.

41 Julian Shun and Guy E. Blelloch. Ligra: a lightweight graph processing framework for shared memory. In *Proceedings of the 18th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP)*, pages 135–146, 2013. `doi:10.1145/2442516.2442530`.

42 Supreme Court of the United States. Ricci v. DeStefano (Nos. 07-1428 and 08-328), 530 F. 3d 87, reversed and remanded. `https://www.law.cornell.edu/supct/html/07-1428.ZO.html`, 2009.

43 Yotaro Takazawa, Shinji Mizuno, and Tomonari Kitahara. Approximation algorithms for the covering-type k-violation linear program. *Optimization Letters*, 13:1515–1521, 2019. `doi:10.1007/S11590-019-01425-W`.

44 The College Board. 2025 total group SAT suite of assessments annual report, 2025. URL: `https://reports.collegeboard.org/media/pdf/2025-total-group-sat-suite-of-assessments-annual-report.pdf`.

45 Géza Tóth. Point sets with many k-sets. *Discrete & Computational Geometry*, 26(2):187–194, 2001. `doi:10.1007/S004540010022`.

46 Akrivi Vlachou, Christos Doulkeridis, Yannis Kotidis, and Kjetil Nørvåg. Reverse top-k queries. In *2010 IEEE 26th International Conference on Data Engineering (ICDE)*, pages 365–376, 2010. `doi:10.1109/ICDE.2010.5447890`.

47 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the international congress of mathematicians: Rio de janeiro 2018*, pages 3447–3487, 2018.

48 Xingxing Xiao and Jianzhong Li. rkHit: Representative query with uncertain preference. *Proceedings of the ACM on Management of Data (SIGMOD)*, 1(2):1–26, 2023. `doi:10.1145/3589271`.

49 Ke Yang, Vasilis Gkatzelis, and Julia Stoyanovich. Balanced ranking with diversity constraints. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6035–6042, 2019. `doi:10.24963/IJCAI.2019/836`.

50 Ke Yang and Julia Stoyanovich. Measuring fairness in ranked outputs. In *Proceedings of the 29th international conference on scientific and statistical database management (SSDBM)*, pages 1–6, 2017. `doi:10.1145/3085504.3085526`.

51 Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM)*, pages 1569–1578, 2017. `doi:10.1145/3132847.3132938`.

52 Meike Zehlike, Ke Yang, and Julia Stoyanovich. Fairness in ranking, part I: Score-based ranking. *ACM Computing Surveys*, 55(6):1–36, 2022. `doi:10.1145/3533379`.

The Squishy Grid Problem

Zixi Cai
IIIS, Tsinghua University, Beijing, China

Kuowen Chen
IIIS, Tsinghua University, Beijing, China

Shengquan Du
IIIS, Tsinghua University, Beijing, China

Arnold Filtser
Bar-Ilan University, Ramat Gan, Israel

Seth Pettie
University of Michigan, Ann Arbor, MI, USA

Daniel Skora
University of Michigan, USA

Abstract

In this paper we consider the problem of approximating Euclidean distances by the infinite integer grid graph. Although the topology of the graph is fixed, we have control over the edge-weight assignment $w : E \to \mathbb{R}_{\geq 0}$, and hope to have grid distances be asymptotically isometric to Euclidean distances, that is:

For all grid points u, v, $\text{dist}_w(u, v) = (1 \pm o(1))\|u - v\|_2$.

We give three methods for solving this problem, each attractive in its own way.

- Our first construction is based on an embedding of the recursive, non-periodic *pinwheel tiling* of Radin and Conway [15, 16, 9] into the integer grid. Distances in the pinwheel graph are asymptotically isometric to Euclidean distances, but no explicit bound on the rate of convergence was known. We prove that the multiplicative distortion of the pinwheel graph is $(1+1/\Theta(\log^{\xi} \log D))$, where D is the Euclidean distance and $\xi = \Theta(1)$. The pinwheel tiling approach is conceptually simple, but can be improved quantitatively.
- Our second construction is based on a hierarchical arrangement of *highways*. It is simple, achieving stretch $(1+1/\Theta(D^{1/9}))$, which converges doubly exponentially faster than the pinwheel tiling approach.
- The first two methods are deterministic, with rigorous guarantees. An even simpler approach is to sample the edge weights independently and randomly from a common distribution $\mathscr{D}$. Whether there exists a distribution $\mathscr{D}^*$ that makes grid distances Euclidean, asymptotically and in expectation, is major open problem in the theory of *first passage percolation*. Previous experiments show that when $\mathscr{D}$ is a Fisher distribution (which is continuous), grid distances are within 1% of Euclidean distances. We demonstrate experimentally that this level of accuracy can be achieved by a simple 2-point distribution that assigns weights 0.41 or 4.75 with probability 44% and 56%, respectively.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases grid graph, Euclidean distance, metric embedding, first passage percolation

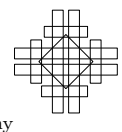
Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.27

Related Version *Full Version*: https://arxiv.org/pdf/2507.23105 [7]

Funding Supported by NSF Grants CCF-2221980 and CCF-2446604 and Israel Science Foundation Grant No. 1042/22.



© Zixi Cai, Kuowen Chen, Shengquan Du, Arnold Filtser, Seth Pettie, and Daniel Skora;
licensed under Creative Commons License CC-BY 4.0





1 Introduction

In this paper we consider a natural geometric problem tangentially related to metric embeddings, spanners, and, in its randomized form, percolation theory. Suppose we wish to approximate Euclidean distances between points on the plane, but with a simple *discrete* structure: the integer grid graph $\mathsf{Grid} = (\mathbb{Z} \times \mathbb{Z}, \{\{u, v\} \mid \|u - v\|_1 = 1\})$. If we consider all edges of $E(\mathsf{Grid})$ to have unit length, then Grid can be regarded as a $\sqrt{2}$-spanner since for any $(u, v) \in (\mathbb{Z}^2)^2$,

$$\|u - v\|_2 \leq \mathrm{dist}_{\mathsf{Grid}}(u, v) \leq \sqrt{2} \cdot \|u - v\|_2.$$

Now define $\mathsf{Grid}[w]$ to be Grid endowed with a non-negative edge-weight assignment $w : E(\mathsf{Grid}) \to \mathbb{R}_{\geq 0}$, and let dist_w be the distance function with respect to w. We consider the natural question: does there exist a $\mathsf{Grid}[w^*]$ that is an asymptotic 1-spanner of the Euclidean plane?

▶ **Question 1** (The Squishy Grid Problem)**.** Does there exist a weight function w^* such that for all $u, v \in V(\mathsf{Grid})$ dist_{w^*} is asymptotically Euclidean? That is,

$$\mathrm{dist}_{w^*}(u, v) = (1 \pm o(1))\|u - v\|_2.$$

If so, we may distinguish various types of convergence:
Polynomial. $\mathrm{dist}_{w^*}(u, v) = \|u - v\|_2 \pm O(\|u - v\|_2)^{1-\Omega(1)}$.
Subpolynomial. $\mathrm{dist}_{w^*}(u, v) = \|u - v\|_2 \pm (\|u - v\|_2)^{o(1)}$.
Constant. $\mathrm{dist}_{w^*}(u, v) = \|u - v\|_2 \pm O(1)$.

Before discussing our approach to answering Question 1 we review the history of Question 1 and its connections to percolation theory.

1.1 History of the Problem and Related Results

G. Tardos (personal communication) made us aware of a 1990 book chapter of Pach, Pollack, and Spencer [14] who attributed some version of Question 1 to Paul Erdős. Pach et al. [14] proved that for any fixed $\epsilon > 0$ there exists a graph $G[\epsilon]$ on the vertex set $\mathbb{Z} \times \mathbb{Z}$ such that for all u, v,

$$\|u - v\|_2 \leq \mathrm{dist}_{G[\epsilon]}(u, v) \leq (1 + \epsilon)\|u - v\|_2 + O(5^{1/\epsilon}).$$

Unfortunately, $G[\epsilon]$ is not planar and obviously depends on ϵ, so it does not lead to a resolution of Question 1. Burago and Ivanov [6] exhibited a regular, weighted graph G on the vertex set $\mathbb{Z} \times \mathbb{Z}$ for which

$$\|u - v\|_2 \leq \mathrm{dist}_G(u, v) \leq \|u - v\|_2 + C,$$

for some absolute constant C. However, G is also not planar.

Borradaile and Eppstein [4] considered a more general problem: given a point set $P \in \mathbb{R}^2$, compute a weighted planar graph $G = (P \cup S, E)$ with *Steiner points* S such that $\mathrm{dist}_G(u, v)$ $(1 + \epsilon)$-approximates the Euclidean distance $\|u - v\|_2$. They proved that $|S| = O_{\epsilon,\alpha}(|P|)$ suffices, where α is the sharpest angle in the Delaunay triangulation of P. A result of Chang, Krauthgamer, and Tan [8] implies an upper bound of $O_\epsilon(|P| \operatorname{polylog} |P|)$, which is slightly superlinear but independent of α.

The problem was first posed to us by G. Bodwin, not as a deterministic design problem (Question 1) but as a *randomized* one. Whenever $\mathscr{D}$ is a distribution over $\mathbb{R}_{\geq 0}$, let $\mathsf{Grid}[\mathscr{D}]$ be the distribution of weighted graphs such that for each $e \in E(\mathsf{Grid})$, $w(e) \sim \mathscr{D}$ is sampled independently from the distribution. Is it possible to find a distribution $\mathscr{D}^*$ such that distances in $\mathsf{Grid}[\mathscr{D}^*]$ are Euclidean in expectation? In more detail:

▶ **Question 2** (Randomized Squishy Grid Problem)**.** Does there exist a distribution $\mathscr{D}^*$ over $\mathbb{R}_{\geq 0}$ such that if $\mathsf{Grid}[w] \sim \mathsf{Grid}[\mathscr{D}^*]$ is a randomly weighted graph, for all $u, v \in V(\mathsf{Grid})$,

$$\mathbb{E}(\mathrm{dist}_w(u,v)) = (1 \pm o(1))\|u - v\|_2.$$

The randomized process implicit in Question 2 is actually not new, but dates back to at least a 1965 paper of Hammersley and Welsh [11], who called it *first passage percolation*. They imagined an *orchard* in which trees were planted on the integer lattice. One tree is initially infected, and the time taken for an infected tree to infect a cardinal neighbor is governed by a distribution $\mathscr{D}$ on $\mathbb{R}_{\geq 0}$. One can then ask: how far does the infection spread by time t? and what does the set of infected trees look like?

Many basic questions in first passage percolation theory remain open, and we can quickly summarize the known facts related to Question 2. Let $\mathbf{0} = (0,0)$ be the origin and e_θ be the unit vector with angle θ degrees. We interpret ne_θ to mean the integer point in $V(\mathsf{Grid})$ nearest to ne_θ. The *time constant* $\mu_0(\mathscr{D})$ is such that $\lim_{n\to\infty} \mathrm{dist}_w(\mathbf{0}, ne_0)/n = \mu_0$ almost surely, which exists if, whenever $w_1, \ldots, w_4 \sim \mathscr{D}$ are independently sampled, $\mathbb{E}(\min\{w_1, w_2, w_3, w_4\}) < \infty$ [12]. It follows that $0 \leq \mu_0 \leq \mathbb{E}(w_1 \sim \mathscr{D})$, with the latter inequality holding with equality only if $w_1 \sim \mathscr{D}$ is constant almost surely [11]. Similarly, the time constants for other angles $\mu_\theta(\mathscr{D}) = \lim_{n\to\infty} \mathrm{dist}_w(\mathbf{0}, ne_\theta)/n$ exist, and collectively define the limiting *shape* of the balls under distribution $\mathscr{D}$. Let $B(t) = \{u \in \mathbb{Z}^2 \mid \mathrm{dist}_w(\mathbf{0}, u) \leq t\}$ be the ball of radius t around the origin. The Cox-Durrett shape theorem [10] shows that with probability 1, as $t \to \infty$, $B(t)/t$ tends to a fixed *limit shape* $\mathcal{B}(\mathscr{D}) \subset \mathbb{R}^2$. When $\mu_0(\mathscr{D}) > 0$, $\mathcal{B}(\mathscr{D})$ is bounded, convex, and has the same symmetries as $\mathbb{Z}^2$, and when $\mu_0(\mathscr{D}) = 0$, $\mathcal{B}(\mathscr{D})$ is $\mathbb{R}^2$ itself. See [2] for an extensive survey of first passage percolation theory.

In the context of answering Question 2 we can rescale any non-trivial distribution $\mathscr{D}$ so that its time constant $\mu_0(\mathscr{D}) = 1$, i.e., distances from the origin to points on the x- and y-axes are asymptotically isometric. In light of the Cox-Durrett theorem, Question 2 asks whether there exists a $\mathscr{D}$ for which $\mathcal{B}(\mathscr{D})$ is the unit L_2 ball $\{x \mid \|x\|_2 \leq 1\}$.

Unfortunately, there are no results characterizing $\mathcal{B}(\mathscr{D})$ for *any* non-trivial distribution $\mathscr{D}$. It is not even known whether there exists $\mathscr{D}$ such that

$$\lim_{n\to\infty} \frac{\mathbb{E}(\mathrm{dist}_w(\mathbf{0}, ne_0))}{n} = \lim_{n\to\infty} \frac{\mathbb{E}(\mathrm{dist}_w(\mathbf{0}, ne_{45}))}{n} = 1, \tag{1}$$

i.e., $\mathcal{B}(\mathscr{D})$ coincides with the unit L_2-ball on the eight (inter)cardinal directions. On the other hand, we have solid experimental evidence that $\mathcal{B}(\mathscr{D})$ can get within 1% of the unit L_2-ball, for certain distributions $\mathscr{D}$. An experimental study of Alm and Deijfen [1] looked at various continuous distributions $\mathscr{D}$. When $\mathscr{D}$ is the uniform distribution, the limit shape $\mathcal{B}(\mathscr{D})$ approximates the L_2-ball to with 4%, whereas when $\mathscr{D}$ is exponential the limit shape is about 1.5% away from the L_2-ball. The best empirical approximation to the L_2-ball came from a Fisher distribution, with error less than 1%.

1.2 Results and Findings

We provide two approaches to answering Question 1 and present experimental evidence that Question 2 can be answered in the affirmative, using simple discrete distributions.

Our first construction is based on Radin and Conway's *pinwheel tiling* [15, 16, 9], a conceptually simple tiling that emerges from the observation that a right triangle with proportions $1 : 2 : \sqrt{5}$ can be partitioned into five right triangles with the same proportions. It is known [16] that when regarded as a plane graph G_{PW} with edges weighted according to Euclidean distance, distances in the pinwheel tiling are Euclidean *in the limit*, that is,

$$\lim_{d\to\infty} \max_{u,v:\|u-v\|>d} \frac{\mathrm{dist}_{G_{\mathrm{PW}}}(u,v)}{\|u-v\|_2} = 1.$$

However the rate of convergence is unknown. We embed the pinwheel tiling into the grid graph, and prove a bound on its convergence, namely that for a constant $\xi = \Theta(1)$,

$$\mathrm{dist}_{G_{\mathrm{PW}}}(u,v) = \left(1 + O\left(\frac{1}{\log^{\xi}\log\|u-v\|_2}\right)\right)\|u-v\|_2.$$

A natural problem is to optimize the *convergence* rate of the construction. We give a new, simple construction of a weight function w of the grid that is asymptotically Euclidean, with a ***polynomial*** convergence rate, namely: $\mathrm{dist}_w(u,v) = \|u-v\|_2 + O(\|u-v\|_2^{8/9})$.

The construction is based on laying out "highways" in the plane, which are paths cleaving closely to a line with a certain slope a, whose edge weights are equal and chosen to approximate Euclidean distances along the highway. For example, when $a \in [0,1]$, the weights are $\frac{\sqrt{a^2+1}}{a+1}$. In order to get a $(1+o(1))$-distance approximation, it is necessary that the set of slopes of all highways be dense in $[0,\pi)$. Thus, there are infinitely many slopes, and infinitely many parallel highways of each slope, whose intersection pattern is quite complicated. The tricky part in the design stage is to decide what to do with intersecting highways. We give a simple method that eliminates intersections while guaranteeing polynomial convergence.

Alm and Deijfen's [1] experimental study of first passage percolation selected $\mathscr{D}$ from various continuous distributions such as uniform, exponential, Gamma, and Fisher distributions. For several of these distributions the limit shape $\mathcal{B}(\mathscr{D})$ approximated the L_2-ball within a few percent, with a Fisher distribution being the best. Our experiments show that very simple distributions with support size 2 or 3 can replicate the accuracy of the continuous distributions [1]. For example, the improbable distribution $\mathscr{D}_2$:

$$\Pr_{w_0\sim\mathscr{D}_2}\left(w_0 = \begin{Bmatrix} 0.41401 \\ 4.75309 \end{Bmatrix}\right) = \begin{Bmatrix} 0.44273 \\ 0.55727 \end{Bmatrix}$$

empirically approximates the Euclidean metric to within about 0.75%, and a certain 3-point distribution $\mathscr{D}_3$ approximates it to within 0.622%. A very natural question is whether other L_p metrics can be approximated, in expectation, by various distributions. We illustrate that the uniform distribution and some Gamma distributions approximate L_p metrics with $p < 2$. None of our experiments support the possibility that L_p metrics with $p > 2$ can be approximated.

1.3 Organization

We present the construction based on pinwheel tilings in Section 2, as well as new bounds on the convergence of the stretch of the pinwheel graph. The highway construction is presented in Section 3, having polynomial convergence. We present the experimental findings in Section 4 and conclude with a discussion of several open problems related to Questions 1 and 2 in Section 5. Sections 2, 3, and 4 are written to be independent.

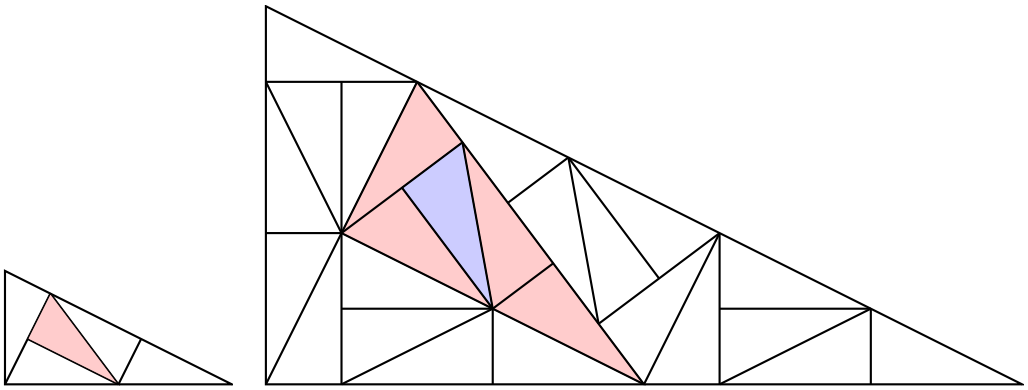


Figure 1 **(A)** $\mathscr{T}_1$, containing $\mathscr{T}_0$ in red. **(B)** $\mathscr{T}_2$, containing $\mathscr{T}_1$ in red, and $\mathscr{T}_0$ in blue.

2 A Deterministic Construction Based on Pinwheel Tilings

The "pinwheel" tiling of Radin [15] is an example of a non-periodic tiling using a single tile type (and its reflection). Let $\mathscr{T}_0, \mathscr{T}_1, \mathscr{T}_2, \ldots$ be a series of tilings of ever larger triangular swatches of the plane, and let $\mathscr{T}_\omega$ be the tesselation of the plane achieved in the limit. $\mathscr{T}_0$ consists of a single right triangle with side lengths $1, 2, \sqrt{5}$. In general $\mathscr{T}_{i+1}$ is formed from $\mathscr{T}_i$ by taking four additional copies of $\mathscr{T}_i$, suitably reflected, rotated, and translated, so that they form a larger triangle with the same $1 : 2 : \sqrt{5}$ proportions. Figure 1 illustrates the construction of $\mathscr{T}_2$ from $\mathscr{T}_1$ and $\mathscr{T}_0$.

2.1 Pinwheel Tilings

By construction $\mathscr{T}_\omega$ is a tiling of the plane using *atomic* triangles with side lengths $1, 2, \sqrt{5}$. Due to the recursive nature of the construction, we can also regard $\mathscr{T}_\omega$ as a tiling using $\sqrt{5}^i, 2\sqrt{5}^i, \sqrt{5}^{i+1}$ triangles, for any integer $i \geq 0$. Observe from Figure 1 that the boundary of $\mathscr{T}_{i+1}$ is obtained from the boundary $\mathscr{T}_i$ by scaling by $\sqrt{5}$, translation, and rotation by $\arctan(1/2)$. We will henceforth define $\gamma = \arctan(1/2)$. As $\gamma/(2\pi)$ is irrational, the orientation of tiles in $\mathscr{T}_\omega$ is uniformly distributed in $[0, 2\pi)$. Radin and Sadun [16] used this fact to prove an isoperimetric property of pinwheel tilings, namely that there are finite subsets of tiles from $\mathscr{T}_\omega$ whose area/perimeter^2 is arbitrarily close to that of the circle. Suppose we regard $\mathscr{T}_\omega$ as a plane graph G_ω, whose vertices and edges are the union of the vertices and edges of all atomic triangles. Radin and Sadun [16] proved that for $u, v \in V(G_\omega)$, $\mathrm{dist}_{G_\omega}(u, v) = (1 + o(1))\|u - v\|_2$. Although the multiplicative stretch is 1 *in the limit*, their proof implies no particular rate of convergence. We prove the following.

▶ **Theorem 1.** *Let G_ω be the plane graph of the pinwheel tiling $\mathscr{T}_\omega$, whose edges are weighted according to the Euclidean distance between their endpoints. Then for any $u, v \in V(G_\omega)$,*

$$\|u - v\|_2 \leq \mathrm{dist}_{G_\omega}(u, v) \leq (1 + O(1/(\log\log\|u - v\|_2)^\xi)) \cdot \|u - v\|_2, \qquad \text{for some } \xi > 0.$$

2.2 Distribution of Tile Orientations

If T is a triangle in the recursive tiling with dimensions $\sqrt{5}^i, 2\sqrt{5}^i, \sqrt{5}^{i+1}$, we call T a *level-i* triangle. As a first step toward proving Theorem 1, we analyze the orientations of the triangles contained within a single large triangle. Given a triangle T of level x, let $\mathrm{Ang}(T, k)$ denote the set of angles attained by the hypotenuses of all level-$(x - k)$ triangles contained within T. We observe that the elements of this set are characterized by an arithmetic recurrence.

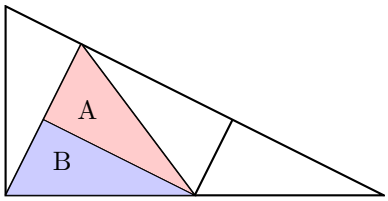


Figure 2 Two triangles within T. The hypotenuses of T, A, B have angles $-\gamma, -2\gamma, 0$, respectively.

▶ **Observation 2.** *Suppose a triangle T of level x has its hypotenuse at angle θ. For $0 \le k \le x$, $\mathrm{Ang}(T,k) \supseteq \{\theta + (2t-k)\gamma \,|\, t \in \{0,\dots,k\}\}$.*

Proof. We proceed by induction on k, with the base case $k = 0$ being trivial. Consult Figure 2, where two triangles A and B of level-$(x-1)$ are depicted. Observe that the hypotenuse of A (resp. B) is rotated by an angle of $-\gamma$ (resp. $+\gamma$) relative to that of the exterior triangle. By the inductive hypothesis,

$$\mathrm{Ang}(A,k-1) \supseteq \{\theta - \gamma + (2t-k+1)\gamma \mid t \in [0,k-1]\} = \{\theta + (2t-k)\gamma \mid t \in [0,k-1]\}$$
$$\mathrm{Ang}(B,k-1) \supseteq \{\theta + \gamma + (2t-k+1)\gamma \mid t \in [0,k-1]\} = \{\theta + (2t-k)\gamma \mid t \in [1,k]\},$$

and the union of these sets is exactly $\{\theta + (2t-k)\gamma \,|\, t \in \{0,\dots,k\}\}$. ◀

Consider some $\lambda \in \mathbb{R}$, and denote by $\{\lambda\} = \lambda - \lfloor\lambda\rfloor$ the fractional part of λ. It is well known due to Weyl's criterion that if λ is irrational, the set $S(\lambda, N) = \{\{kx\} : 1 \le k \le N\}$ becomes uniformly distributed in $[0,1)$ as $N \to \infty$ [5]. Moreover, the rate of convergence is controlled by the *irrationality exponent* $\mu(\lambda)$ which measures the asymptotic quality of rational approximations to λ. For a real number λ, $\mu(\lambda)$ is defined to be the supremum of the set of real numbers μ such that $0 < \left|\lambda - \frac{p}{q}\right| < \frac{1}{q^\mu}$ has infinitely many solutions for integers p, q with $q > 0$. A finite irrationality exponent implies that a number does not have a sequence of rational approximations that are "too good." For sets $S \subseteq U \subseteq \mathbb{R}$, we say S is an *ε-cover* of U if for every $x_1 \in U$ there exists $x_2 \in S$ satisfying $|x_1 - x_2| < \varepsilon$. When $\mu(\lambda)$ is finite, $S(\lambda, N)$ is an ε-cover of $[0,1)$ for $N = \varepsilon^{1-\mu(\lambda)-o(1)}$ [13]. The following corollary follows by applying this fact to Observation 2.

▶ **Corollary 3.** *Given a triangle T of level-x, $\mathrm{Ang}(T,k)$ is a θ-cover of $[0,2\pi)$ for $k = \theta^{1-\mu(\gamma/\pi)-o(1)}$.*

Before bounding the irrationality exponent $\mu(\gamma/\pi)$, we quickly review some terminology related to the algebraic numbers $\overline{\mathbb{Q}}$. For $\alpha \in \overline{\mathbb{Q}}$, its *minimal polynomial* is the unique polynomial $P \in \mathbb{Z}[x]$ of lowest degree with relatively prime coefficients such that $P(\alpha) = 0$. We say the *degree* of α is the degree of P, while the *height* of α is the absolute value over coefficients of P.

Both γ and π can be expressed in the form $\beta \ln \alpha$ where $\alpha \in \overline{\mathbb{Q}}$ and $\beta \in \mathbb{Q}(i)$. Since $e^{i\gamma} = \frac{2+i}{\sqrt{5}}$, $\gamma = -i \ln(\frac{2+i}{\sqrt{5}})$. Similarly, π can be written as $-i \ln(-1)$. A great deal of work in the mid-20th century yielded various lower bounds on linear forms in logarithms of algebraic numbers. A history of the problem up to 1976 can be found in [3]. For our purposes, we choose a simple bound due to Baker.

▶ **Theorem 4** (Baker [3])**.** *Suppose for $n \ge 1$ we have algebraic numbers $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n \in \overline{\mathbb{Q}} \setminus 0$. If the logarithms $\ln \alpha_i$ are linearly independent over the rational numbers, then*

$$|\beta_1 \ln \alpha_1 + \cdots + \beta_n \ln \alpha_n| > H^{-C},$$

where H is the maximum of the heights of the β_i and C is a function of n, the numbers α_i, and the degrees of the numbers β_i.

Consider arithmetic expressions of the form $\beta_1 \ln \alpha_1 + \beta_2 \ln \alpha_2$ where $\alpha_1, \alpha_2 \in \overline{\mathbb{Q}} \setminus 0$ are fixed and $\beta_1, \beta_2 \in \mathbb{Z} \setminus 0$. Now β_1 and β_2 have degree 1, so the exponent C becomes a constant, and $H = \max\{|\beta_1|, |\beta_2|\}$. Applying Baker's theorem (Theorem 4) and dividing both sides by $|\beta_1 \ln \alpha_2|$ yields $\left|\frac{\ln \alpha_1}{\ln \alpha_2} + \frac{\beta_2}{\beta_1}\right| > \frac{1}{|\beta_1 \ln \alpha_2| H^C} \geq \frac{1}{|\ln \alpha_2| H^{C+1}}$. By choosing $\alpha_1 = \frac{2+i}{\sqrt{5}}$ and $\alpha_2 = -1$, the ratio $\frac{\ln \alpha_1}{\ln \alpha_2}$ is exactly γ/π. Recall that this number is irrational, satisfying the criteria that $\ln \alpha_1$ and $\ln \alpha_2$ are linearly independent over the rationals. Since $\gamma/\pi < 1$, clearly we can replace H by β_1 to obtain $\left|\frac{\ln \alpha_1}{\ln \alpha_2} + \frac{\beta_2}{\beta_1}\right| > \frac{1}{|\ln \alpha_2| \beta_1^{C+1}}$. Comparing this with the definition of the irrationality exponent, this is sufficient to see that $\mu(\gamma/\pi)$ is at most $C+1$ and therefore finite. Henceforth let $\mu = \mu(\gamma/\pi)$. It is worth noting that the constant C given by Theorem 4 is effectively computable, though the order of magnitude is impractical.[1]

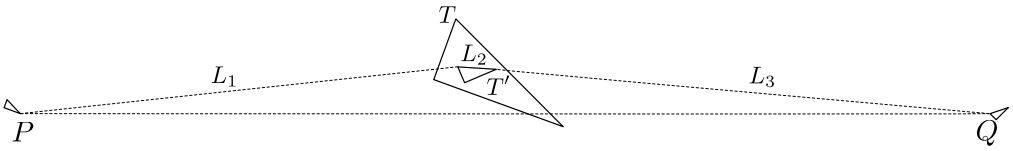
2.3 Convergence of Stretch

We define $f(d)$ to be the maximum stretch guaranteed by G_ω, over all pairs of vertices at Euclidean distance at least d.

$$f(d) = \sup \left\{ \frac{\text{dist}_{G_\omega}(u, v)}{\|u - v\|_2} \;\middle|\; u, v \in V(G_\omega) \text{ and } \|u - v\|_2 \geq d \right\}.$$

▶ **Lemma 5.** *Fix any distance d and let $f(d) = 1+\varepsilon$. There exists a function $n = \varepsilon^{(1-\mu)/2-o(1)}$ such that if $d = \Omega(5^n/\varepsilon)$, then $f(3d) < 1 + \varepsilon - \Omega(5^{-n})$.*

Proof. Consider any two vertices $P, Q \in G_\omega$ with $|\overline{PQ}| = D \geq 3d$ when we regard them as points in the plane. Begin by choosing a parameter $\delta = \delta(\varepsilon)$ and identifying a triangle T satisfying the following properties: T intersects $\overline{PQ}$, the projection of T onto $\overline{PQ}$ lies entirely within the middle third of $\overline{PQ}$, and the hypotenuse of T has length exactly δD. Now choose a parameter $\theta = \theta(\varepsilon)$ and let $n = n(\theta)$ be large enough such that $\text{Ang}(T, n)$ is a θ-cover of $[0, \pi)$. Finally, select a triangle T' within T, and exactly n levels below T, whose hypotenuse creates an angle less than θ with $\overline{PQ}$.



We now construct a path as follows. Let p and q be the endpoints of the hypotenuse of T' closer to P and Q respectively. Let $L_1 = \overline{Pp}$, $L_2 = \overline{pq}$, and $L_3 = \overline{Qq}$. $|L_1|$ and $|L_3|$ must each be at least d, so by the definition of the stretch function f, G_ω approximates L_1 and L_3 to within $1 + \varepsilon$ stretch.

Let ℓ_1, ℓ_2, and ℓ_3 be the projections of L_1, L_2, and L_3 onto $\overline{PQ}$. We claim that $\text{dist}_{G_\omega}(P, p)/|\ell_1| < (1 + \varepsilon)(1 + 5\delta^2)$. Since p lies within T, its projection onto $\overline{PQ}$ is a distance at most δD from p and at least $D/3$ from P. Therefore, $|L_1|/|\ell_1|$ is bounded by $\sqrt{(D/3)^2 + (\delta D)^2}/(D/3) = \sqrt{1 + 9\delta^2} < 1 + 5\delta^2$. By definition of f, $\text{dist}_{G_\omega}(P, p)/|L_1| \leq 1 + \varepsilon$.

[1] One estimate, also due to Baker [3], states that $C = \ln A_1 \ln^2 A_2 (16nd)^{200n}$ suffices when the β_i are rational, where A_i is the height of α_i and d is the degree of the field extension $\mathbb{Q}[\alpha_1, \alpha_2]/\mathbb{Q}$. For our purposes, $A_1 = 4$, $A_2 = 2$, $n = 2$, and $d = 4$, giving $C \approx 2^{2800}$. Baker remarks it is possible to argue C is much lower in reality, but the required analysis is quite technical and beyond the scope of this paper.

The same is true for $\text{dist}_{G_\omega}(q,Q)/|\ell_3|$. On the other hand, $\text{dist}_{G_\omega}(p,q)/|\ell_2|$ is trivially bounded by $|L_2|/|\ell_2| \le 1/\cos\theta$ since p and q are joined directly by an edge in G_ω. Combining the bounds on each component, we write

$$\text{dist}_{G_\omega}(P,Q) < (1+\varepsilon)(1+5\delta^2)(|\ell_1|+|\ell_3|) + \frac{|\ell_2|}{\cos\theta}.$$

We normalize by $1/D$ to obtain the stretch, and because this inequality holds for arbitrary P, Q satisfying $|\overline{PQ}| \ge 3d$, it holds for the supremum as well.

$$\begin{aligned} f(3d) &= \sup_{P,Q\in V(G_\omega)} \left\{ \frac{\text{dist}_{G_\omega}(P,Q)}{|\overline{PQ}|} \;\middle|\; |\overline{PQ}| \ge 3d \right\} \\ &< (1+\varepsilon)(1+5\delta^2)\left(1 - \frac{|\ell_2|}{D}\right) + \frac{|\ell_2|}{D\cos\theta} \\ &\le (1+\varepsilon)(1+5\delta^2)(1-5^{-n/2}\delta\cos\theta) + 5^{-n/2}\delta \end{aligned}$$

Letting $\kappa = 5^{-n/2}$, this is upper bounded by

$$< 1 + \varepsilon + 6\delta^2 + \kappa\delta(1-(1+\varepsilon)\cos\theta)$$

At this point we fix $\theta = \sqrt{\varepsilon}$, so $\cos\theta > 1 - \varepsilon/2$.

$$< 1 + \varepsilon + 6\delta^2 - \kappa\delta\varepsilon(1-\varepsilon)/2 = 1 + \varepsilon + \delta(6\delta - \kappa\varepsilon(1-\varepsilon)/2)$$

Finally, we pick $\delta < \kappa\varepsilon/24$.

$$\le 1 + \varepsilon - \Omega(\delta^2) = 1 + \varepsilon - \Omega(5^{-n}).$$

By Corollary 3, we choose $n(\theta) = n(\sqrt{\varepsilon}) = (\sqrt{\varepsilon})^{1-\mu-o(1)}$. The only remaining detail is to address that the triangles T and T' used in this argument do indeed exist. The length of the hypotenuse of T was taken to be δD, and then we chose T' to be n levels beneath that of T. Therefore, we have implicitly assumed $D \ge 5^{n/2}/\delta > \Omega(5^n/\varepsilon)$, consistent with the assumption in the statement of the lemma. ◀

Theorem 1 now follows easily from this lemma; see the full version [7].

2.4 Pinwheel Tilings on the Grid

▶ **Theorem 6.** *There exists a weight function w such that* Grid$[w]$ *has stretch 1, asymptotically. In particular, for any $u, v \in V(\text{Grid})$,*

$$\|u-v\|_2 - O(1) \le \text{dist}_w(u,v) \le (1+O(1/(\log\log\|u-v\|_2)^\xi))\cdot\|u-v\|_2, \quad \textit{for some } \xi > 0.$$

The proof of this theorem can be found in the full version [7].

3 A Deterministic Construction with Faster Convergence

In this section, we give a deterministic construction based on *highways* with faster convergence. Specifically, we establish the following result as Theorem 7:

▶ **Theorem 7.** *There exists an assignment $W : E(\text{Grid}) \to \mathbb{R}_{\ge 0}$ such that for any $u, v \in V(\text{Grid})$,*

$$\|u-v\|_2 - 1 \le \text{dist}_W(u,v) \le \|u-v\|_2 + O\left(\|u-v\|_2^{\frac{8}{9}}\right).$$

We prove Theorem 7 in two steps. First, we show that the same statement holds for the finite square grid $[n] \times [n]$ (Theorem 8), then we give a black-box reduction from the infinite case to the finite case.

▶ **Theorem 8.** *There exists a weight assignment $W(n)$ to edges of the finite grid on $[n] \times [n]$ such that for any $u, v \in [n]^2$,*

$$\|u - v\|_2 - 1 \leq \text{dist}_{W(n)}(u, v) \leq \|u - v\|_2 + O\left(\|u - v\|_2^{\frac{8}{9}}\right).$$

The proof of Theorem 8 follows from the constructions presented in Section 3.1 and Section 3.2.

3.1 Highways

Given a line $\ell = ax + b$ in $\mathbb{R}^2$ we define the Highway(ℓ) to be a grid-path that tracks ℓ; see Figure 3.

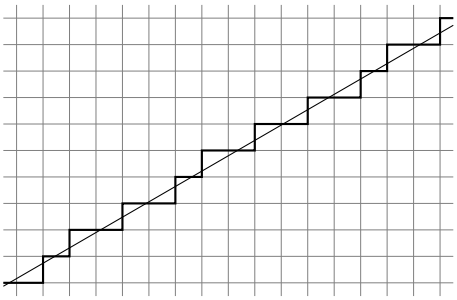


Figure 3 A grid-path that tracks ℓ.

Specifically, let $v \in V(\mathsf{Grid})$ be a grid-point, and consider the 1×1 square $[v-0.5, v+0.5) \times [v-0.5, v+0.5)$. If ℓ intersects this square, then we include v in $V(\ell)$. Whenever $u, v \in V(\ell)$ are adjacent grid points, Highway(ℓ) contains the edge $\{u, v\} \in E(\mathsf{Grid})$. If $a \in [-1, 1]$ we let w_ℓ assign every edge in Highway(ℓ) the weight $\frac{\sqrt{a^2+1}}{|a|+1}$, which is the asymptotic ratio between the Euclidean distance of points on ℓ and the number of edges taken along the grid path Highway(ℓ). Otherwise we write ℓ as $x = a^{-1}(y - b)$ with $a^{-1} \in [-1, 1]$ and use weight $\frac{\sqrt{a^{-2}+1}}{|a^{-1}|+1}$. All off-highway edges have weight ∞.

This weight assignment guarantees a discrepancy of at most 1 between Euclidean distances and grid distances along Highway(ℓ). The proof of Lemma 9 appears in [7].

▶ **Lemma 9.** *Let* Highway(ℓ) *denote the highway that approximates a line ℓ of the form $y = ax + b$. Then for any two points $u, v \in V(\text{Highway}(\ell))$, $|\text{dist}_{w_\ell}(u, v) - \|u - v\|_2| \leq 1$.*

The highway transformation can also be applied to a line *segment* s. We use the same notation Highway(s).

3.2 The Hierarchical Highway Construction

We are trying to find a weight assignment for the finite grid $[n] \times [n]$ in order to prove Theorem 8. The construction is parameterized by $(k_i)_{1 \leq i \leq m}$, where $k_1 = \lfloor n^{1/5} \rfloor$, $\quad k_{i+1} = \lfloor k_i^{1/2} \rfloor$, and m is minimum such that $k_m < 100$.

The construction is based on a hierarchical system of lines in $\mathbb{R}^2$ which will eventually be embedded as highways in the grid. The lines at level i have angles selected from $(\theta_{i,j})_{0\le j<k_i}$, where $\theta_{i,j} = \frac{\pi \cdot j}{k_i}$. Fixing i and one such angle $\theta_{i,j}$, there are many lines with angle $\theta_{i,j}$, spaced at distance k_i^4. For $i \in [1,m], j \in [0, k_i - 1], t \in \mathbb{Z}$, define

$$\ell_{i,j,t} = \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; y \cdot \cos\theta = x \cdot \sin\theta + t \cdot k_i^4\right\}.$$

Define $\mathsf{Lines}[i] = \{\ell_{i,j,t}\}$ to be the set of all lines at level i.

We cannot choose a weight function $W(n)$ that agrees with $w_{\ell_{i,j,t}}$ for *every* line $\ell_{i,j,t}$ due to intersections. Our solution is to avoid this issue by removing all line intersections, which introduces distortions in distances that must be bounded.

Below we define a procedure to remove parts of $\ell_{i,j,t}$, leaving a set of line *segments* $\mathcal{L}_{i,j,t}$. Define $\mathsf{Segments}[i] = \bigcup \mathcal{L}_{i,j,t}$ to be the set of all line segments at level i. If O is an object or collection of objects, define

$$\mathrm{Fat}(O,\delta) = \{p \mid \exists q \in O \text{ such that } \|p-q\|_2 \le \delta\}$$

to be all points within distance δ of O. Specifically, $\mathrm{Fat}(\ell,\delta)$ is a strip if ℓ is a line, and a hippodrome if ℓ is a segment.

Once $\mathsf{Segments}[1], \ldots, \mathsf{Segments}[i-1]$ are constructed, we construct $\mathsf{Segments}[i]$ as follows. For each $\ell_{i,j,t} \in \mathsf{Lines}[i]$ initialize $\ell \leftarrow \ell_{i,j,t}$ and proceed to remove parts of ℓ in Steps 1 and 2.

Step 1. Set $\ell \leftarrow \ell - \mathrm{Fat}(\mathsf{Lines}[i] - \{\ell\}, k_i)$, i.e., we remove every part of ℓ within distance k_i of any other line at level i.

Step 2. For each segment $s \in \bigcup_{i'<i} \mathsf{Segments}[i']$ such that $\mathrm{Fat}(s,k_i) \cap \ell \neq \emptyset$, let $\overline{AB} = \mathrm{Fat}(s,k_i) \cap \ell$. If $\|A-B\|_2 \ge k_i$, set $\ell \leftarrow \ell - \overline{AB}$. Otherwise, let $B' \in \ell_{i,j,t}$ be such that $\|A-B'\| = k_i$ and $B \in \overline{AB'}$, and set $\ell \leftarrow \ell - \overline{AB'}$. See Figure 4. Define $\mathcal{L}_{i,j,t} = \ell$, and include all line segments of $\mathcal{L}_{i,j,t}$ in $\mathsf{Segments}[i]$.

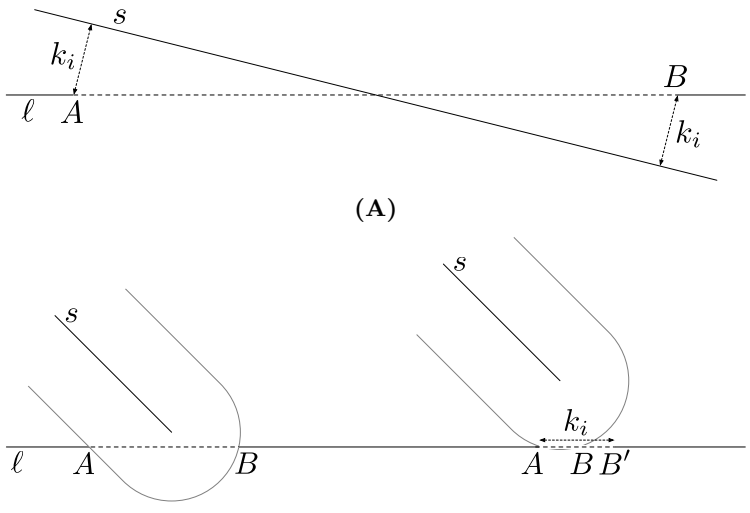


Figure 4 Illustrations of various cases in Step 2. **(A)** ℓ and s intersect. The segment $\ell \cap \mathrm{Fat}(s,k_i)$ is removed from ℓ. **(B)** ℓ and s do not intersect. Left: $\overline{AB} \ge k_i$ and $\overline{AB}$ is removed from ℓ. Right: $\overline{AB} < k_i$ and B' is such that $\overline{AB'} = k_i$ is removed from ℓ.

The weight assignment $W(n)$ is now constructed as follows. For each line segment $s \in \bigcup_i \mathsf{Segments}[i]$, let $W(n)$ agree with w_s at all edges in the corresponding highway segment $\mathrm{Highway}(s)$. All edges not appearing in any line segment have weight 2.

▶ **Lemma 10.** *Fix any $s \in \mathsf{Segments}[i]$, $s' \in \mathsf{Segments}[i']$, where $s \neq s'$ and $i \geq i'$. For any points $u \in s, u' \in s'$, $\|u - v\|_2 \geq k_i$.*

Proof. Steps 1 and 2 ensure that all remaining points on $\mathcal{L}_{i,j,t}$ lie outside

$$\mathrm{Fat}\left(\bigcup_{i'<i} \mathsf{Segments}[i'] \cup (\mathsf{Lines}[i] - \{\ell_{i,j,t}\}), k_i\right),$$

and that any two consecutive segments on $\mathcal{L}_{i,j,t}$ are separated by distance at least k_i. This implies that the Euclidean distance between any $u \in s$ and $u' \in s'$ is at least k_i. ◀

Lemma 11 shows that distances under $W(n)$ are approximately Euclidean, up to a multiplicative stretch of $1 + O(k_i^{-1})$ and additive stretch $O(k_i^4)$, for every index i.

▶ **Lemma 11.** *For any $u, v \in [n]^2$ and $i \in [m]$.*

$$\|u - v\|_2 - 1 \leq \mathrm{dist}_{W(n)}(u, v) \leq \|u - v\|_2 + O\left(k_i^{-1}\|u - v\|_2 + k_i^4\right).$$

Proof. Observe that the highways corresponding to all segments in $\bigcup_i \mathsf{Segments}[i]$ are vertex-disjoint. Thus, every grid path P from u to v can be written as $B_0A_1B_1A_2B_2\cdots A_kB_k$, where each $A_j \subset \mathrm{Highway}(s)$ for some $s \in \bigcup_i \mathsf{Segments}[i]$ and each B_j is disjoint from all highway segments, and therefore consists of only weight-2 edges. (One or both of B_0, B_k may be empty.) By Lemma 9, $W(A_j)$ is at least the Euclidean distance between its endpoints minus one, and $W(B_j)$ is at least the Euclidean distance multiplied by 2, which implies that $W(B_j)$ is at least the Euclidean distance plus 1 for $1 \leq j \leq k-1$, so $W(P)$ is at least $\|u - v\|_2 - 1$.

Turning to the upper bound, We bound the distance $\mathrm{dist}_{W(n)}(u, v)$ by explicitly constructing a path that stays within the vicinity of a single line $\ell_{i,j,t}$. Since level-i lines occur at angular intervals of $\frac{\pi}{k_i}$ and parallel lines are spaced k_i^4 apart, we can always find an $\ell^* = \ell_{i,j,t}$ satisfying the following properties. First, the difference in angle between ℓ^* and $\overline{uv}$ is at most $\frac{\pi}{2k_i}$. Second, the distance from u to ℓ^* is at most $k_i^4/2$. Let A and B be the closest grid points on $\mathrm{Highway}(\ell^*)$ from u and v respectively. It follows that

$$\mathrm{dist}_{W(n)}(u, A) + \mathrm{dist}_{W(n)}(v, B) = O\left(k_i^4 + \|u, v\|_2 \sin\left(\frac{\pi}{2k_i}\right)\right) = O(k_i^4 + k_i^{-1}\|u - v\|_2). \quad (2)$$

It remains to bound $\mathrm{dist}_{W(n)}(A, B)$. A trivial upper bound is $\mathrm{dist}_{W(n)}(A, B) \leq 2\cdot\sqrt{2}\|A - B\|_2$, so if $\|A - B\|_2 < 100k_i^4$ we are done. Henceforth we shall assume that $\|A - B\|_2 \geq 100k_i^4$. We would prefer to follow the A-B path along $\mathrm{Highway}(\ell^*)$, but sections of this highway have effectively been removed by Steps 1 and 2 of the construction. We bound the stretch induced by the gaps in the highway introduced in Steps 1 and 2 separately.

Step 1 Stretch. Fix a direction $\theta_{i,j'}$ different from ℓ^*'s direction $\theta_{i,j}$. Whenever ℓ^* intersects a line with angle $\theta_{i,j'}$, Step 1 causes $\mathrm{Highway}(\ell^*)$ to lose $\frac{k_i}{\sin|\theta_{i,j} - \theta_{i,j'}|} = O(k_i^2)$ edges. There are k_i angles, and parallel lines with angle $\theta_{i,j'}$ are spaced k_i^4 apart, so the total number of edges removed from the A-B path in $\mathrm{Highway}(\ell^*)$ in Step 1 is

$$O\left(k_i^2 \cdot k_i \cdot \frac{\|A - B\|_2}{k_i^4}\right) = O(k_i^{-1}\|A - B\|_2). \quad (3)$$

The additive stretch induced by walking across the gaps induced by Step 1 is also $O(k_i^{-1}\|A - B\|_2)$ as all these edges have weight 2.

Step 2 Stretch. Whenever part of Highway(ℓ^*) is removed by Step 2 we do *not* walk precisely in the direction of ℓ^* but take a *detour* to a lower level highway. Suppose that in Step 2, a segment $s \in \bigcup_{i'<i} \mathsf{Segments}[i']$ causes an interval $\overline{CD}$ of ℓ^* to be removed. Define E, F to be the points on s closest to C, D, respectively. When our path reaches C, we walk from C to E, then to F along Highway(s), then to D. See Figure 5. By construction $\|C-E\|_2, \|D-F\|_2 = O(k_i)$. Since $\|E-F\|_2 \le \|C-D\|_2$ and by Lemma 9 $\mathrm{dist}_{W(n)}(E,F) = \|E-F\|_2 \pm 1$, the additive stretch due to the conflict with s is at most

$$\mathrm{dist}_{W(n)}(C,E) + \mathrm{dist}_{W(n)}(E,F) + \mathrm{dist}_{W(n)}(F,D) - \mathrm{dist}_{w_{\ell^*}}(C,D) = O(k_i).$$

The last task is to bound the number of such segments interfering with the A-B path. Observe that s is a segment of a line at level $i-1$ or lower. Thus, by Lemma 10 any two such segments s, s' are at distance at least $k_{i-1} \ge k_i^2$, and the total additive stretch caused by Step 2 detours is

$$O\left(k_i \cdot \frac{\|A-B\|_2}{k_i^2}\right) = O(k_i^{-1}\|A-B\|_2). \tag{4}$$

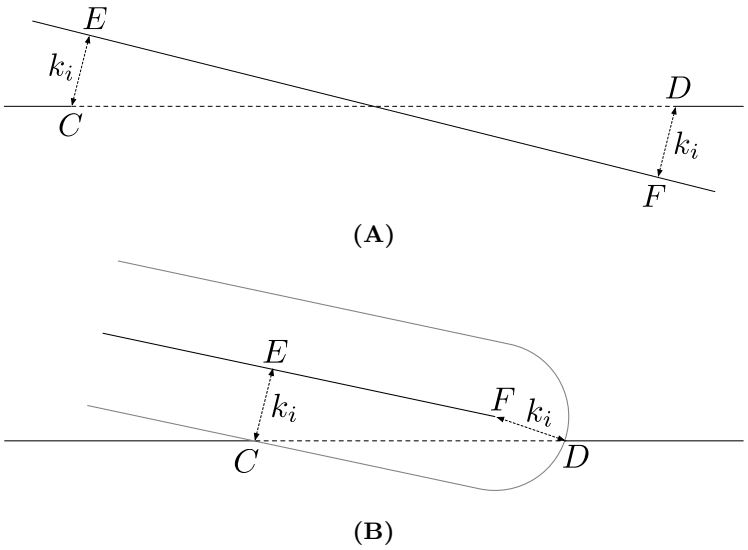


Figure 5 Step 2 detours from the proof of Lemma 11. **(A)** The case when segment s intersects ℓ^*. **(B)** When segment s does not intersect ℓ^*, but $\mathrm{Fat}(s, k_i)$ does.

Combining Equations (2)–(4), we conclude that

$$\begin{aligned}\mathrm{dist}_{W(n)}(u,v) &\le \mathrm{dist}_{W(n)}(u,A) + \mathrm{dist}_{W(n)}(A,B) + \mathrm{dist}_{W(n)}(B,v)\\ &\le \|u-v\| + O\left(k^{-1}\|A-B\|_2 + k_i^{-1}\|u-v\|_2 + k_i^4\right)\\ &= \|u-v\|_2 + O\left(k^{-1}\|u-v\|_2 + k_k^4\right).\end{aligned}$$

◀

Proof of Theorem 8. Recall that by the definition of the sequence (k_i), for any pair of points u, v with Euclidean distance $d = \|u-v\|_2 > 100^9$, there exists some index $1 \le i \le m$ such that $k_i \in \left[d^{1/9}-1, d^{2/9}\right]$. Applying Lemma 11, we have

$$\mathrm{dist}_W(u,v) \le \|u-v\|_2 + O\left(\frac{1}{k_i}\cdot\|u-v\|_2 + k_i^4\right) \;\le\; \|u-v\|_2 + O\left(\|u-v\|_2^{8/9}\right).$$

The theorem holds trivially when $d \le 100^9$, which completes the proof of Theorem 8. ◀

In the full version [7] we give a black-box reduction from any finite construction on $[n]\times[n]$, like that of Theorem 8, to a construction on $\mathbb{Z}\times Z$, yielding Theorem 7.

4 Experimental Findings in the Squishy Grid

If Question 2 seems too daunting, a natural idea is to simplify the problem by considering only *monotone* paths, that is, paths that use the fewest number of edges.

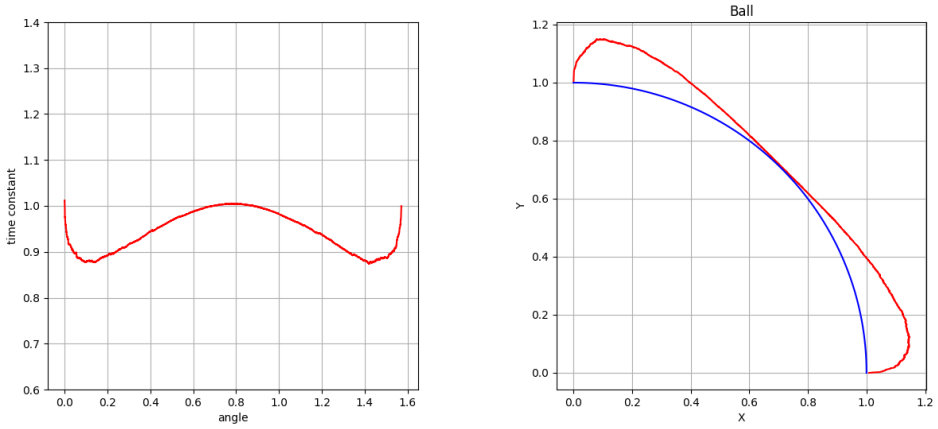
▶ **Question 3.** Define $\text{monodist}_w(u,v)$ to be the length of the shortest u-v path that uses $\|u-v\|_1$ edges. Is there a distribution $\mathscr{D}^*$ over $\mathbb{R}_{\geq 0}$ such that if $\mathsf{Grid}[w] \sim \mathsf{Grid}[\mathscr{D}^*]$ is a randomly weighted graph, for all $u,v \in V(\mathsf{Grid})$, $\mathbb{E}(\text{monodist}_w(u,v)) = (1 \pm o(1))\|u-v\|_2$?

At first glance this problem may seem easier, or more plausible, than Question 2. Whereas it is an open problem finding a distribution $\mathscr{D}$ satisfying Equation (1) (the time constant in the 0° and 45° directions are 1), this is nearly trivial when we consider monodist. Refer to [7] for proof.

▶ **Lemma 12.** *There exists a distribution $\mathscr{D}$ on $\mathbb{R}_{\geq 0}$ such that*

$$\lim_{n\to\infty} \frac{\mathbb{E}(\text{monodist}_w(\mathbf{0}, ne_0))}{n} = \lim_{n\to\infty} \frac{\mathbb{E}(\text{monodist}_w(\mathbf{0}, ne_{45}))}{n} = 1.$$

Let $B_{\text{mono}}(t) = \{u \in \mathbb{Z}^2 \mid \text{monodist}_w(\mathbf{0},u) \leq t\}$ and $\mathcal{B}_{\text{mono}}(\mathscr{D})$ be the limiting shape of $B_{\text{mono}}(t)/t$ in $\mathsf{Grid}[w] \sim \mathsf{Grid}[\mathscr{D}]$ as $t \to \infty$. Thus, $\mathcal{B}_{\text{mono}}(\mathscr{D})$ coincides with the L_2-ball in the eight intercardinal directions N, E, S, W, NE, SE, SW, NW. If $\mathcal{B}_{\text{mono}}(\mathscr{D})$ were convex, then it would have to be quite close to the L_2-ball. Unfortunately, our experiments show that $\mathcal{B}_{\text{mono}}(\mathscr{D})$ is *not* convex, which casts serious doubt on Question 3 having an affirmative answer. See Figure 6.



(A) **(B)**

Figure 6 **(A)** The stretch of $\text{monodist}_w(\mathbf{0}, ne_\theta)/\|ne_\theta\|_2$, as a function of the angle $\theta \in [0, \frac{\pi}{2}]$, expressed in radians. **(B)** The shape of $\mathcal{B}_{\text{mono}}(\mathscr{D}_{\epsilon^*})$.

In retrospect, Question 3 is less likely than Question 2 to be answered in the affirmative since $\text{monodist}_w(\mathbf{0},(n,m))$, $m \leq n$, is much more sensitive to small deviations in m than $\text{dist}_w(\mathbf{0},(n,m))$ Considering the cases when $m = 0, m = 0.1n$, and $m = n$, $\text{monodist}_w(\mathbf{0},(n,m))$ is the minimum of $\binom{n+0}{0} = 1$, $\binom{n+0.1n}{0.1n} \approx 1.39^n$, and $\binom{2n}{n} \approx 4^n$ different paths, respectively. This sharp jump from constant to exponential in the vicinity $m = 0$ does not exist in Question 2. Assuming the variance of $\mathscr{D}$ is sufficiently large, $\text{dist}_w(\mathbf{0}, ne_0)$ is the minimum of an exponential number of plausible shortest paths.

4.1 Discrete Distributions for First Passage Percolation

When dealing with discrete distributions the most natural measure of complexity is *support size*. Therefore, we study Question 2 experimentally by considering the space of 2- and 3-point distributions. For a fixed integer k, the k-point distribution $\mathscr{D}(\{(p_i, x_i)\}_{i=1}^k)$, is such that $\Pr_{w_0 \sim \mathscr{D}}(w_0 = x_i) = p_i$. It is determined by $2k-1$ parameters, as $p_k = 1-(p_1+\cdots+p_{k-1})$.

4.1.1 Experimental Methodology

To identify locally optimal distributions among k-point discrete distributions of the form $\mathscr{D}\big(\{(p_i, x_i)\}_{i=1}^k\big)$, we employ a two-layer iterative strategy:

- We first perturb the probability vector $(p_1, \ldots, p_k)$.
- For fixed probabilities $(p_1, \ldots, p_k)$, we generate k random initial values $x_1, x_2, \ldots, x_k$ and then alternate between the following two update steps:

  Perturbation Step. We perturb each value x_i and compute the estimated directional stretch for both $\theta = 0$ and $\theta = \frac{\pi}{4}$. Let μ_θ be the empirical ratio $\frac{\mathrm{dist}_w(\mathbf{0}, ne_\theta)}{n}$, obtained from this round of simulation. Here $n \approx 30{,}000$.

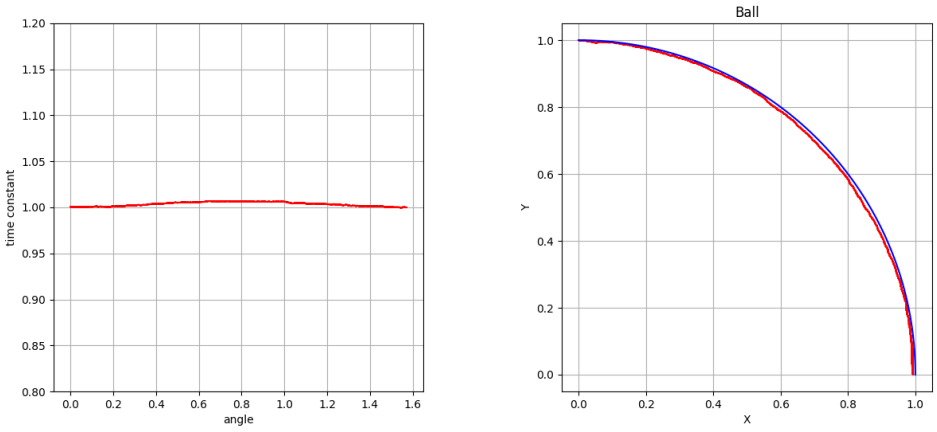
  Normalization Step. We normalize the values $\{x_i\}_{i=1}^k$ by setting $x_i \leftarrow \frac{x_i}{\sqrt{\mu_0 \cdot \mu_{\frac{\pi}{4}}}}$. This scaling ensures that the average stretch along the cardinal and intercardinal directions remains close to 1, thereby facilitating comparisons between distributions.

4.1.2 2-Point Distributions

The best 2-point distribution identified with this method is $\mathscr{D}_2$, given below. Roughly speaking, every edge weight is either 0.41 or 4.75, 44% and 56% of the time, respectively.

$$\mathscr{D}_2 = \{(0.44273, 0.41401), (0.55727, 4.75309)\}.$$

We find that $\mathsf{Grid}[w] \sim \mathsf{Grid}[\mathscr{D}_2]$ empirically approximates Euclidean distances up to stretch 1.00750, i.e., up to (3/4)% error. Figure 7(A) plots the observed stretch $\mathrm{dist}_w(\mathbf{0}, ne_\theta)/\|ne_\theta\|_2$ as a function of the angle $\theta \in [0, \pi/2)$. Figure 7(B) shows the set of all grid points whose empirical graph distance from the origin first exceeds n. The resulting boundary is visually close to a Euclidean circle, suggesting that $\mathscr{D}_2$ induces an approximately isotropic metric in expectation.



(A) **(B)**

Figure 7 Results on the 2-point distribution $\mathscr{D}_2$.**(A)** Stretch $\mathrm{dist}_w(\mathbf{0}, ne_\theta)/\|ne_\theta\|_2$, as a function of the angle $\theta \in [0, \frac{\pi}{2}]$. **(B)** The empirical distance-n ball in $\mathsf{Grid}[w] \sim \mathsf{Grid}[\mathscr{D}_2]$.

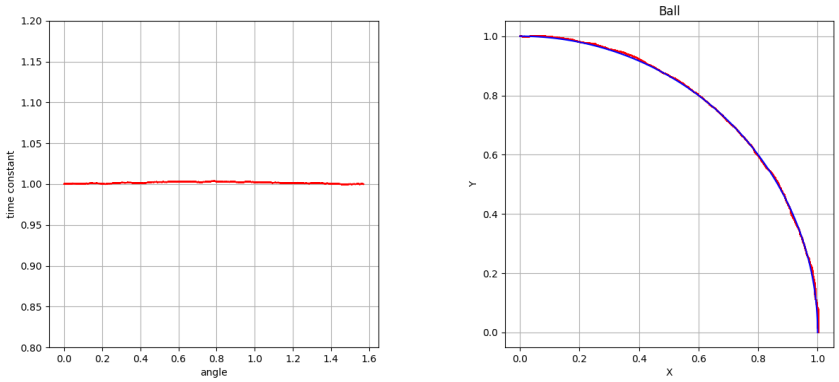
Note that $\mathbb{E}(w_0 \sim \mathscr{D}_2) \approx 2.83$, meaning that $\text{dist}_w(\mathbf{0}, ne_0) \approx n$ is likely to be realized by a highly non-monotone path, consistent with the observations in Figure 6.

4.1.3 3-Point Distributions

$\mathscr{D}_2$ does not leave much room for improvement, but we are able to eke out a slightly better empirical stretch of 1.00622 with a 3-point distribution $\mathscr{D}_3$.

$$\mathscr{D}_3 = \mathscr{D}(\{(0.34809, 0.20647), (0.25735, 2.51586), (0.39456, 9.32215)\}).$$

See Figure 8 for visual representations of the empirical stretch of $\mathscr{D}_3$.



(A) **(B)**

Figure 8 Results on the 3-point distribution $\mathscr{D}_3$. **(A)** Stretch $\frac{\text{dist}_w(\mathbf{0}, ne_\theta)}{\|ne_\theta\|_2}$, as a function of $\theta \in [0, \frac{\pi}{2}]$. **(B)** The empirical distance-n ball in $\mathsf{Grid}[w] \sim \mathsf{Grid}[\mathscr{D}_3]$.

5 Conclusion

In this paper we asked how well the integer grid graph Grid can approximate Euclidean distances, if weighted appropriately. We gave two deterministic weighting schemes that answer Question 1, the best one achieving a ***polynomial*** additive stretch, that is, $\mathbb{E}[\text{dist}_w(u, v)] = \|u - v\|_2 + O(\|u - v\|_2^{1-\delta})$ for $\delta = 1/9$. Improving the additive stretch to something ***subpolynomial*** $\|u - v\|_2^{o(1)}$ seems to require a new approach to the problem. Our "highway" method seems incapable of achieving subpolynomial additive error, and even if Question 2 is answered affirmatively (choosing weights i.i.d. from some distribution $\mathscr{D}^*$), the tail bounds here only give polynomial additive error [2].

We conjecture that no weighting achieves subpolynomial additive error.

▶ **Conjecture 13.** *The additive error* $\text{dist}_w(u, v) - \|u - v\|_2$ *of* $w : E(\mathsf{Grid}) \to \mathbb{R}_{\geq 0}$ *is a function of* $d = \|u - v\|_2$.
Weak Conjecture. *There is no* w *with constant additive error* $O(1)$*, independent of* d.
Strong Conjecture. *There is no* w *with subpolynomial additive error* $d^{o(1)}$.

In the randomized setting, Question 2 is equivalent to an old problem in *first passage percolation* [2, 11] that is unsolved, but has some empirical evidence in its favor. Alm and Deijfen [1] showed that when $\mathscr{D}$ is a certain Fisher distribution, that $\mathsf{Grid}[w] \sim \mathsf{Grid}[\mathscr{D}]$

approximates Euclidean distances to less than 1% error. In this paper, we demonstrated that a simple 2-point distribution $\mathscr{D}_2$ achieves 0.75% error, and that a 3-point distribution $\mathscr{D}_3$ achieves 0.622% error. It is an interesting open problem to prove that in $\mathsf{Grid}[w] \sim \mathsf{Grid}[\mathscr{D}_2]$, the expected error of dist_w is less than 1%.

References

1 Sven Erick Alm and Maria Deijfen. First passage percolation on $\mathbb{Z}^2$: A simulation study. *J. Stat. Phys.*, 161:657–678, 2015. doi:10.1007/s10955-015-1356-0.

2 Antonio Auffinger, Michael Damron, and Jack Hanson. *50 Years of First-Passage Percolation*. University Lecture Series. American Mathematical Society, 2017.

3 Alan M. Baker. The theory of linear forms in logarithms. In *Transcendence theory: advances and applications (A. Baker, D. W. Masser, Eds.)*, pages 1–27. Academic Press, London, 1976.

4 Glencora Borradaile and David Eppstein. Near-linear-time deterministic plane Steiner spanners for well-spaced point sets. *Comput. Geom.*, 49:8–16, 2015. doi:10.1016/J.COMGEO.2015.04.005.

5 Yann Bugeaud. *Distribution Modulo One and Diophantine Approximation*, pages 1–14. Cambridge University Press, 2012.

6 Dmitri Burago and Sergei Ivanov. Uniform approximation of metrics by graphs. *Proc. Amer. Math. Soc.*, 143:1241–1256, 2015.

7 Zixi Cai, Kuowen Chen, Shengquan Du, Arnold Filtser, Seth Pettie, and Daniel Skora. The squishy grid problem. *CoRR*, abs/2507.23105, 2025. doi:10.48550/arXiv.2507.23105.

8 Hsien-Chih Chang, Robert Krauthgamer, and Zihan Tan. Almost-linear ϵ-emulators for planar graphs. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1311–1324, 2022. doi:10.1145/3519935.3519998.

9 John H. Conway and Charles Radin. Quaquaversal tilings and rotations. *Inventiones Mathematicae*, 132:179–188, 1998.

10 J. Theodore Cox and Richard Durrett. Some limit theorems for percolation with necessary and sufficient conditions. *Annals of Probability*, 9:583–603, 1981. doi:10.1214/aop/1176994364.

11 John M. Hammersley and Dominic J. A. Welsh. First-passage percolation, subadditive processes, stochastic networks, and generalized renewal theory. In *Bernoulli 1713, Bayes 1763, Laplace 1813: Anniversary Volume, Proceedings of an International Research Seminar, Statistical Laboratory, University of California, Berkeley, 1963*, pages 61–110. Springer, 1965.

12 Harry Kesten. Aspects of first passage percolation. In *École d'Été de Probabilités de Saint Flour XIV, Lecture Notes in Mathematics, 1180*. Springer, 1986.

13 Lauwerens Kuipers and Harald Niederreiter. *Uniform Distribution of Sequences*, pages 118–132. Wiley, 1974.

14 János Pach, Richard Pollack, and Joel Spencer. Graph distance and Euclidean distance on the grid. In *Topics in Combinatorics and Graph Theory: Essays in Honour of Gerhard Ringel*, pages 555–559. Springer, 1990.

15 Charles Radin. The pinwheel tilings of the plane. *Annals of Mathematics*, 139:661–702, 1994.

16 Charles Radin and Lorenzo Sadun. The isoperimetric problem for pinwheel tilings. *Commun. in Math. Phys.*, 177:255–263, 1996.

Triangulating a Polygon with Holes in Optimal (Deterministic) Time

Timothy M. Chan
Siebel School of Computing and Data Science, University of Illinois at Urbana-Champaign, IL, USA

Abstract

We consider the problem of triangulating a polygon with n vertices and h holes, or relatedly the problem of computing the trapezoidal decomposition of a collection of h disjoint simple polygonal chains with n vertices total. Clarkson, Cole, and Tarjan (1992) and Seidel (1991) gave randomized algorithms running in $O(n \log^* n + h \log h)$ time, while Bar-Yehuda and Chazelle (1994) described deterministic algorithms running in $O(n + h \log^{1+\varepsilon} h)$ or $O((n + h \log h) \log\log h)$ time, for an arbitrarily small positive constant ε. No improvements have been reported since. We describe a new $O(n + h \log h)$-time algorithm, which is optimal and deterministic.

More generally, when the given polygonal chains are not necessarily simple and may intersect each other, we show how to compute their trapezoidal decomposition (and in particular, compute all intersections) in optimal $O(n + h \log h)$ deterministic time when the number of intersections is at most $n^{1-\varepsilon}$.

To obtain these results, Chazelle's linear-time algorithm for triangulating a simple polygon is used as a black box.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Polygons, triangulation, intersection, derandomization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.28

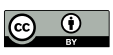
Related Version *Full Version*: https://arxiv.org/abs/2603.21617

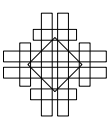
1 Introduction

Triangulating a simple polygon is one of the most basic problems in computational geometry, with numerous applications. Chazelle's deterministic linear-time algorithm [5] represents one of the landmark results in the area, improving previous deterministic $O(n \log\log n)$-time algorithms by Tarjan and Van Wyck [28] and Kirkpatrick, Klawe, and Tarjan [20] and previous randomized $O(n \log^* n)$-time algorithms by Clarkson, Tarjan, and Van Wyck [10] and Seidel [27]. Here, n denotes the number of vertices in the input polygon and $\log^*$ is the (very slow-growing) iterated logarithm function.

In this paper, we study an extension of the problem: triangulating a polygon with h holes (where the holes themselves are disjoint simple polygons enclosed in an outer simple polygon). See Figure 1 (left). The problem can be equivalently stated in a number of different ways:

1. Computing the *trapezoidal decomposition* of a polygon with h holes. The decomposition (also called the "trapezoidization", or "vertical decomposition", or "vertical visibility map") is obtained by adding vertical line segments from each vertex to the edge immediately above/below the vertex. Linear-time reductions from trapezoidal decompositions to triangulations and vice versa are known [14].
2. Computing the trapezoidal decomposition of a collection of h disjoint simple polygonal chains. See Figure 1 (right). (We can enclose the chains with a bounding box and turn each chain into an infinitesimally thin hole, while increasing n and h by only a constant factor.)


© Timothy M. Chan;
licensed under Creative Commons License CC-BY 4.0



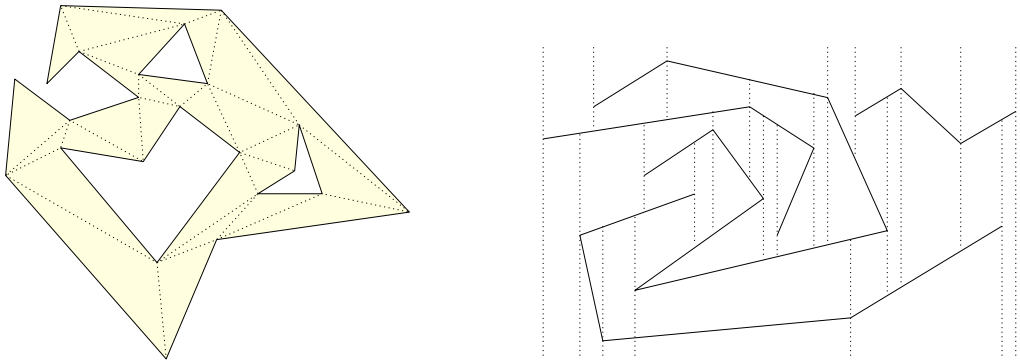


Figure 1 (Left) A triangulation of a polygon with 3 holes. (Right) The trapezoidal decomposition of 5 disjoint simple polygonal chains.

3. Computing the trapezoidal decomposition of a plane straight-line graph (a graph drawn on the plane with straight-line edges without crossings, where edges are ordered around each vertex) with h connected components. (We can make tiny cuts to eliminate cycles and then take an Euler tour to turn each component into a chain, while increasing n and h by only a constant factor.)

Previous randomized $O(n \log^* n)$-time algorithms for simple polygons can be adapted to solve the problem in $O(n \log^* n + h \log h)$ expected time [8, 27]. In 1994, Bar-Yehuda and Chazelle [3] gave deterministic algorithms running in $O(\min\{n + h \log^{1+\varepsilon} h, (n + h \log h) \log\log h\})$ time for an arbitrarily small constant $\varepsilon > 0$. These results came close to the known lower bound of $\Omega(n + h \log h)$,[1] but there is still an extra factor of $\log^* n$ for randomized algorithms, and $\log^{\varepsilon} h$ or $\log\log h$ for deterministic. No improvement has been reported since, deterministic or randomized.

Meanwhile, a number of papers (e.g., [7, 19, 24, 29]) have given algorithms for various problems related to shortest paths and visibility in polygonal domains (polygonal environments with multiple disjoint polygonal obstacles) whose time complexities are dominated by that of the triangulation problem and would be optimal if the triangulation complexity is $O(n + h \log h)$. However, with the current gap of knowledge, these papers needed to state their results more awkwardly, either by assuming that a triangulation of the free space of the polygonal domain (a polygon with holes) is given, or by adding an extra term T to the time bound where T represents the unknown complexity of the triangulation problem. (A similar story occurred for various computational problems about simple polygons in the days before Chazelle's linear-time algorithm.)

Main result. We give the first optimal algorithm for triangulating a polygon with holes in $O(n + h \log h)$ time. The algorithm is deterministic.

Many applications follow [7, 19, 24, 29]. (For example, combining with Wang's result [29] yields an optimal $O(n + h \log h)$-time algorithm for finding the Euclidean shortest path between two given points amidst h disjoint polygonal obstacles with n vertices total.)

[1] An $\Omega(h \log h)$ lower bound in the algebraic decision tree model can be shown by a simple reduction from sorting: to sort h numbers $x_1, \ldots, x_h \in (-1, 1)$, create a polygon that has a triangular outer boundary with vertices $(-1, 1), (1, 1), (0, -1)$, and a sufficiently small hole near (x_i, x_i^2) for each $i \in \{1, \ldots, h\}$.

Previous approaches. Known approaches, as well as our new algorithm, work with the reformulation of the problem in terms of the trapezoidal decomposition of h disjoint polygonal chains.

One could try to directly modify Chazelle's linear-time algorithm for $h = 1$ [5], but this appears difficult (considering how notoriously complicated his algorithm was). Bar-Yehuda and Chazelle's approach was to directly reduce the general h case to the $h = 1$ case: join the multiple polygonal chains into one connected component by adding vertical line segments, and then apply Chazelle's algorithm as a black box. To do so, the first observation is that polygonal chains intersecting a common vertical line can be joined together this way by *Jordan sorting* (the problem of sorting intersections of polygonal chains along a line), which is solvable by known techniques [18]. Roughly, their idea was then to build an *interval tree* [11, 26] for the x-spans of the chains, and solve the problem for each level of the interval tree (since each level consists of vertically separated instances in which chains intersect a common vertical line). The remaining task is merging the results of these $O(\log h)$ levels. Bar-Yehuda and Chazelle proposed a binary divide-and-conquer over the $O(\log h)$ levels, yielding $O((n + h \log h) \log \log h)$ time overall, and alternatively a divide-and-conquer with a larger fan-out, yielding $O(n + h \log^{1+\varepsilon} h)$. This approach seems hard to improve further.

A different approach is randomized divide-and-conquer, as done by Clarkson et al. and Seidel [10, 27]. The trapezoidal decomposition of a random sample of $n/\log n$ edges allows us to divide the problem into subproblems of roughly logarithmic size. Assigning input elements to subproblems requires some form of "point location" and would normally cost $O(\log n)$ per element, but can be done faster knowing that the input elements comes from h chains. Since the recursion has $O(\log^* n)$ levels, the running time inherently increases by a $\log^* n$ factor, unfortunately. Even for $h = 1$ (the case of a simple polygon), Clarkson et al. and Seidel's methods do not achieve linear time (Amato, Goodrich, and Ramos [1] described a randomized linear-time algorithm which is simpler than Chazelle's, but still relies on some of Chazelle's ingredients and is still complicated). Furthermore, randomization seems essential: such random sampling methods where the sample size is large is generally harder to derandomize.

Our approach. The difficulty of improving the random sampling approach turns out to be overstated. We observe that in combination with another known divide-and-conquer strategy via *planar graph separators* [22] or *r-divisions* [15, 16], the random sampling approach can indeed achieve optimal $O(n + h \log h)$ expected running time. The reason the $\log^* n$ factor is avoidable is that we can recurse for a constant number of levels (in fact, just once) and then switch to Chazelle's algorithm (or more precisely Bar-Yehuda and Chazelle's algorithm, which calls Chazelle's) in the base case when the input size is sufficiently small – the $h \log h$ term is big enough to absorb the cost of the switch. The idea of combining sampling with r-divisions and invoking Chazelle's in the base case is not original and can be viewed as a variant of an existing randomized algorithm by Eppstein, Goodrich, and Strash [13] for a related problem (see below). Still, it is interesting that this implication has not been noted earlier, and we include this optimal randomized solution in Appendix A.

Our main contribution is in obtaining an optimal deterministic algorithm. To do so, we combine ideas from both the random sampling approach and Bar-Yehuda and Chazelle's deterministic approach, even though the two approaches appear different and incompatible. To derandomize the sampling approach, we need to compute a subset of the input edges that "behave" like a random sample, known as an *ε-net* in the literature [17, 23] (with ε being $\frac{\text{polylog}\, n}{n}$ in our case). Constructing a net deterministically appears challenging, especially

since we can only afford linear time. We observe a simple way to construct a net in linear time when the trapezoidal decomposition is given, by using planar graph separators or r-divisions (the net size is suboptimal, but is good enough for our purposes).

However, there is an obvious circularity issue: we are not given the trapezoidal decomposition, since that is what we want to compute! Our key idea is to go back to Bar-Yehuda and Chazelle's interval-tree approach, which gives a partition of the input into $O(\log n)$ subsets where we already know how to compute the trapezoidal decomposition of each subset. Although merging their trapezoidal decompositions is not easy, merging their nets is: the union of the nets of the subsets is a net for the whole input (the net size is worsen by an $O(\log n)$ factor, but is still good enough for our purposes). By building the trapezoidal decomposition of the net in combination with an r-division, we can then divide the problem into subproblems of polylogarithmic size. The detailed description of the whole algorithm is given in Section 3, and is just about 2 pages long.

When there are intersections. In the last part of the paper, we address an extension of the problem: computing the trapezoidal decomposition of polygonal chains that may not be simple, i.e., may self-intersect, and may intersect each other. The decomposition is defined in the same way, except that we also add vertical segments from each intersection point to the edge immediately above/below. The crux of the problem lies in the computation of the intersections. (Once all the intersection points are known, we could actually reduce to the non-intersecting case to obtain the final trapezoidal decomposition.)

The problem of computing all self-intersections of one chain (the $h = 1$ case) is already challenging. Chazelle [5] posed the (still open) question of whether there is an $O(n+X)$-time algorithm, where X denotes the number of intersections. Known line-segment intersection algorithms [9, 25, 6, 2] can solve the problem in $O(n \log n + X)$ time, which is optimal when $X \gg n \log n$ but not when X is sublinear; they do not exploit the fact that the input line segments form a connected chain. Clarkson, Cole, Tarjan [8] showed that the random sampling approach yields an $O(n \log^* n + X)$ expected time bound for this problem. On the other hand, Eppstein, Goodrich, and Strash [13] used random sampling in combination with r-divisions to obtain an alternative $O(n + X \log^{(c)} n)$ expected time bound for an arbitrarily large constant c, where $\log^{(c)}$ denotes the logarithm function iterated c times. Thus, their algorithm runs in optimal linear expected time when $X \ll n/\log^{(c)} n$. We are not aware of any *deterministic* result better than $O(n \log n + X)$, however.

We present a deterministic algorithm for computing the trapezoidal decomposition of a non-simple polygonal chain with X intersections in $O(n + Xn^{\varepsilon})$ time. Thus, our algorithm runs in optimal linear time when $X \ll n^{1-\varepsilon}$. (This case is already sufficient for various applications; for example, Eppstein and Goodrich [12] provided empirical evidence that road networks tend to have $X \approx \sqrt{n}$ intersections.)

More generally, when there are multiple chains ($h > 1$), Clarkson, Cole, and Tarjan [8] obtained an $O(n \log^* n + h \log h + X)$ expected time bound. Our deterministic algorithm, presented in Section 4, takes $O(n + h \log h + Xn^{\varepsilon})$ time. (In Appendix A, we also note an extension of Eppstein, Goodrich, and Strash's randomized algorithm with $O(n + h \log h + X \log^{(c)} n)$ expected time bound.)

2 Preliminaries

We begin with some standard definitions.

Trapezoidal decomposition. For a set S of n line segments in the plane, the *trapezoidal decomposition*, denoted $TD(S)$, is the subdivision of the plane formed by S together with vertical line segments drawn from each vertex v of the arrangement (i.e., each endpoint and

each intersection point) to the segment immediately above and below v. The faces in this subdivision are indeed trapezoids (or triangles) where the left and right sides (if exist) are vertical and the top and bottom sides (if exist) are parts of the input segments of S. If S is disjoint, $TD(S)$ has $O(n)$ trapezoids and can be constructed in $O(n \log n)$ time [11, 26]. If S has X intersections, $TD(S)$ has $O(n+X)$ trapezoids and can be constructed in $O((n+X) \log n)$ time by the classic Bentley–Ottmann sweep [4, 26] or in $O(n \log n + X)$ time by the fastest known algorithms [9, 25, 6].

The trapezoidal decomposition of a collection P of (not necessarily simple) polygonal chains, denoted $TD(P)$, is defined as the trapezoidal decomposition of its edge set.

r-division. We use the following weak definition of r-divisions [15]: For a planar graph G with n vertices, an *r-division* is a subset B of $O(n/\sqrt{r})$ vertices such that every connected component in $G - B$ has size at most r. An r-division can be viewed as an extension of planar graph separators, and can be constructed in $O(n)$ time by known algorithms [16, 21] for any given r.

In our application, we consider an r-division of the dual[2] of a trapezoidal decomposition. Here, B is a subset of trapezoids. Construct the subdivision of the plane formed by the sides of the trapezoids in B. The cells of this subdivision will be referred to as the *regions* of the r-division in this paper. Each region is a polygon possibly with holes, which is the union of at most r trapezoids (a region could also just be a single trapezoid, for example, when it is from B). The total number of sides in all the regions[3] is $O(n/\sqrt{r})$. (With a stronger version of r-divisions [15, 21], it is possible to ensure that there are $O(n/r)$ regions, each having at most $O(\sqrt{r})$ sides and at most $O(1)$ holes, though this will not be necessary for our application.)

Interval tree. For a set $\mathcal{I}$ of h intervals, the *interval tree* [11, 26] is a binary tree defined recursively as follows. If $\mathcal{I} = \emptyset$, the tree is just a single node. Otherwise, let a, b, m be the minimum, maximum, and median endpoint. At the root, store $[a, b]$, which we call the *range*, and m, which we call the *pivot value*. At the root, also store all intervals of $\mathcal{I}$ containing m. Recursively build the interval tree for the subset of all intervals of $\mathcal{I}$ completely left of m (this is the left subtree) and the interval tree for the subset of all intervals of $\mathcal{I}$ completely right of m (this is the right subtree). The resulting tree has $O(\log h)$ height, and can be constructed in $O(h \log h)$ time. Note that each interval of $\mathcal{I}$ is stored in exactly one node.

3 Computing the trapezoidal decomposition of h disjoint simple polygonal chains

We now present our main deterministic algorithm for computing the trapezoidal decomposition of h disjoint simple polygon chains (from which an algorithm for triangulating a polygon with h holes would follow).

▶ **Theorem 1.** *Given a collection P of h disjoint simple polygonal chains with a total of n vertices, we can compute $TD(P)$ in $O(n + h \log h)$ deterministic time.*

[2] In the dual graph, the vertices are the trapezoids, and two trapezoids are adjacent if their boundaries (vertical sides or parts of input line segments) overlap.

[3] We use sides instead of edges to avoid confusion between geometric and graph terminologies.

Proof. Let s and t be parameters to be set later. Our algorithm is described in a series of steps below. The purpose of Steps 0–2 is to compute a subset R satisfying a certain property. Afterwards, Steps 3–5 use its trapezoidal decomposition $TD(R)$ to divide the problem into smaller subproblems and then solve these subproblems.

0. For each chain $p \in P$, let I_p be the interval determined by the smallest and largest x-coordinate in p; these can be generated in $O(n)$ time. Build the interval tree for these $O(h)$ intervals in $O(h \log h)$ time. For each node v of the tree, let P_v be the list of all chains $p \in P$ whose intervals I_p are stored at v. Note that each chain appears in exactly one list.

1. For each node v of the interval tree with pivot value m_v:

a. Sort the intersections of P_v with the vertical line $x = m_v$ (note that all chains in P_v intersect this line). This is an instance of the *Jordan sorting* problem for multiple disjoint polygonal chains. Bar-Yehuda and Chazelle [3] showed that this can be done in $O(n_v + h_v \log h_v)$ time (by adapting a known Jordan sorting algorithm for a single polygonal chain [18]), where n_v and h_v are the number of vertices and chains in P_v respectively. Note that $\sum_v n_v = n$ and $\sum_v h_v = h$.

b. Join all chains in P_v into a single simple, connected component P'_v by adding vertical line segments between all pairs of consecutive intersection points along $x = m_v$, by using the output of step 1(a). (Not all such vertical segments are needed to connect the chains, as shown in Figure 2 (left), but this would not matter.) Apply Chazelle's algorithm [5] to compute $TD(P'_v)$, and thus $TD(P_v)$, in $O(n_v)$ time.

c. Compute an s-division of the dual of $TD(P_v)$ in $O(n_v)$ time. Preprocess the regions of the s-division for point location [26] in linear time. Let R_v be the subset of all $O(n_v/\sqrt{s})$ edges of P_v that participate in the region boundaries in this s-division (this excludes vertical sides). Then R_v obeys the following property:[4]

> $(*)$ Given any vertical line segment e not intersecting R_v, the segment e can intersect at most $O(s)$ edges of P_v; furthermore, these edges can be found in $O(\log n + s)$ time.

This property is easy to see: any such segment e must lie completely inside a region in the s-division, which can be found by point location [26] in $O(\log n)$ time, and there are at most $O(s)$ edges of P_v participating in the region.

2. Let R be the union of all the R_v's, together with $O(h)$ extra line segments of zero length at the start vertices of the chains of P. Note that $|R| = O(n/\sqrt{s} + h)$, and R satisfies the following property:[5]

> $(\dagger)$ Given any vertical line segment e not intersecting R, the segment e can intersect at most $O(s \log h)$ edges of P; furthermore, these edges can be found in $O((\log n + s) \log h)$ time.

This follows immediately from property $(*)$, since the x-coordinate of a vertical segment e can lie in the ranges of only $O(\log h)$ nodes v of the interval tree.

---

[4] This property basically says that R_v is an $O(s/n_v)$-*net* [17, 23] of the edge set of P_v, in the set system corresponding to edges intersecting vertical line segments. The size of our net ($O(n_v/\sqrt{s})$) isn't the smallest possible (which would be $O(n_v/s)$) but is good enough for our purposes.

[5] This property basically says that R is an $O((s \log h)/n)$-net of the edge set of P (ignoring the zero-length segments).

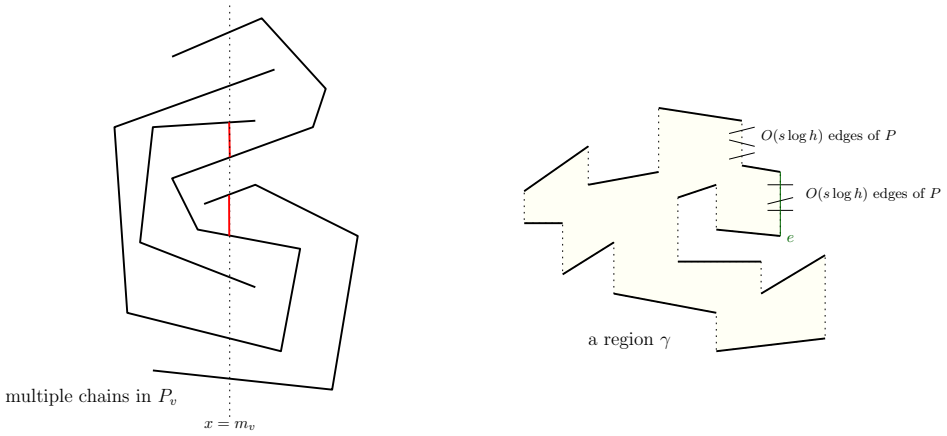


Figure 2 (Left) In Step 1b, we can join multiple chains in P_v intersecting a vertical line $x = m_v$ into a single component, for example, by adding the red vertical segments shown. (Right) In Step 4, we find the $O(s \log h)$ edges of P intersecting each vertical side e of a region γ; the non-vertical sides are parts of edges of R. Each region γ is the union of $O(t)$ trapezoids of $TD(R)$, and the total number of sides over all regions γ is $O(|R|/\sqrt{t})$.

3. Compute $TD(R)$ in $O(|R| \log |R|)$ time, and compute a t-division of the dual of $TD(R)$ in $O(|R|)$ time.

4. For each region γ in the t-division and each vertical side e of the boundary $\partial\gamma$, compute all intersections of e with P. Since the interior of e does not intersect R, we know that e has at most $O(s \log h)$ intersections with P by property (†), and they can be found in $O((\log n + s) \log h)$ time. (See Figure 2 (right).) By summing over all $O(|R|/\sqrt{t})$ choices of e, there are at most $O((|R|/\sqrt{t}) \cdot s \log h)$ such intersections in total and they can be found in $O((|R|/\sqrt{t}) \cdot (\log n + s) \log h)$ time.

   Cut the chains of P at all these intersection points to obtain a new collection P' of polygonal chains. In order to cut, we may need to sort the intersections along each edge of P, but this costs at most a logarithmic factor per intersection, i.e., $O((|R|/\sqrt{t}) \cdot s \log h \cdot \log n)$ in total. The number of vertices in P' is $n' \le n + O((|R|/\sqrt{t}) \cdot s \log h)$, and the number of chains in P' is $h' \le h + O((|R|/\sqrt{t}) \cdot s \log h)$.

5. For each region γ in the t-division, let P'_γ be the chains of P' inside γ, and compute $TD(P'_\gamma)$ by applying Bar-Yehuda and Chazelle's previous algorithms [3] in $O(n'_\gamma + h'_\gamma \log^{O(1)} h'_\gamma)$ time, where n'_γ and h'_γ are the number of vertices and chains in P'_γ respectively.[6] Note that $\sum_\gamma n'_\gamma = n'$ and $\sum_\gamma h'_\gamma = h'$. Furthermore, $\max_\gamma h'_\gamma \le O(t \cdot s \log h)$ (since there are at most $O(t)$ sides in γ, and at most $O(t)$ start vertices of chains inside γ due to those extra segments of zero length added to R).

   Afterwards, we can piece together $TD(P'_\gamma)$ (after clipping to γ) over all regions γ, to obtain $TD(P)$ in $O(n')$ additional time.

The total running time for Steps 0–1 is $O(n + h \log h)$. The running time for Step 3 is $O(|R| \log |R|) = O((n/\sqrt{s} + h) \log n)$ time, as $|R| = O(n/\sqrt{s} + h)$. The total running time for Steps 4–5 is at most $O(n + ((|R|/\sqrt{t}) \cdot s \log h \cdot \log n) + (h + (|R|/\sqrt{t}) \cdot s \log h) \log^{O(1)}(t \cdot s \log h))$.

[6] We could avoid Bar-Yehuda and Chazelle's algorithm and use some slower algorithm running in $O(n'_\gamma + (h'_\gamma)^{O(1)})$ time, if we bootstrap one more time.

Setting $s = \log^2 n$ and $t = \log^8 n$ gives an overall time bound of $O(n + h \log n + h(\log\log n)^{O(1)}) = O(n + h \log n)$, which is the same as $O(n + h \log h)$ (since the second term dominates only if $h \geq n/\log n$, but then $\log n = O(\log h)$). ◀

As a corollary, we can test whether a collection P of h polygonal chains has any intersection in $O(n + h \log h)$ deterministic time: simply run the above algorithm to attempt to compute $TD(P)$, and run Chazelle's linear-time algorithm [5] to check simplicity of the resulting (connected) diagram.

4 When there are a sublinear number of intersections

In this section, we consider the more general case when the polygonal chains may self-intersect or intersect each other, with sublinear number of intersections. Extra ideas are needed. In particular, the interval-tree idea is no longer applicable, and edges may now cross non-vertical sides of the boundary of a region.

In the following lemma, we begin with a special case where we have b subcollections of chains with no intersections within each subcollection.

▶ **Lemma 2.** *Let P be a collection of h simple polygonal chains with a total of n vertices. Suppose P is partitioned into b subcollections $P_1, \ldots, P_b$ such that there are no intersections within each subcollection P_i. Then we can compute $TD(P)$ in $O(n + h \log h + b^{O(1)} X \log^{O(1)} n)$ deterministic time, where X is the number of intersections.*

Proof. Let ℓ be the maximum number of vertices per chain in P. Let s and t be parameters to be set later. We describe an algorithm below to compute all intersections of P. Steps 1–2 compute a subset R satisfying a certain property. Afterwards, Steps 3–5 use its trapezoidal decomposition $TD(R)$ to divide the problem into smaller subproblems and then solve these subproblems.

1. For each $i \in \{1, \ldots, b\}$:
   a. Apply Theorem 1 to compute $TD(P_i)$ in $O(n_i + h_i \log h_i)$ time, where n_i and h_i are the number of vertices and chains in P_i respectively.
   b. Compute an s-division of $TD(P_i)$ in $O(n_i)$ time. Preprocess the regions of the s-division for point location [26] in linear time. Let R_i be the subset of all $O(n_i/\sqrt{s})$ edges of P_i that participate in the boundaries of regions in this s-division.
2. Let R be the union of all the R_i's, together with $O(h)$ extra line segments of zero length at the start vertices of the chains of P. Note that $|R| = O(n/\sqrt{s} + h)$, and R satisfies the following property, similar to before:

   (†) Given any vertical line segment e not intersecting R, the segment e can intersect at most $O(bs)$ edges of P; furthermore, these edges can be found in $O(b(\log n + s))$ time.
3. Compute $TD(R)$ in $O(|R| \log |R| + X)$ time, and compute a t-division of $TD(R)$ in $O(|R| + X)$ time.
4. For each region γ in the t-division and each vertical side e of the boundary $\partial\gamma$, compute all intersections of e with P. Since the interior of e does not intersect R, there are at most $O(bs)$ such intersections and they can be found in $O(b(\log n + s))$ time by property (†). By summing over all $O((|R| + X)/\sqrt{t})$ choices of e, there are at most $O(((|R| + X)/\sqrt{t}) \cdot bs)$ such intersections in total and they can be found in $O(((|R| + X)/\sqrt{t}) \cdot b(\log n + s))$ time.

Cut the chains of P at all these intersection points to obtain a new collection P' of polygonal chains. In order to cut, we may need to sort the intersections along each edge of P, but this costs at most a logarithmic factor per intersection, i.e., $O(((|R|+X)/\sqrt{t}) \cdot bs \cdot \log n)$ in total. The number of vertices in P' is $n' \le n + O(((|R|+X)/\sqrt{t}) \cdot bs)$, and the number of chains in P' is $h' \le h + O(((|R|+X)/\sqrt{t}) \cdot bs)$.

5. For each region γ in the t-division, let P'_γ be the set of all chains of P' with start vertices in γ, let Q_γ be the set of all non-vertical sides of $\partial\gamma$, and apply Theorem 1 to test whether $P'_\gamma \cup Q_\gamma$ has any intersection in $O(n'_\gamma + (h'_\gamma + |Q_\gamma|)\log n)$ time, where n'_γ and h'_γ are the number of vertices and chains in P'_γ respectively. Note that $\sum_\gamma n'_\gamma = n'$ and $\sum_\gamma h'_\gamma = h'$ and $\sum_\gamma |Q_\gamma| = O((|R|+X)/\sqrt{t})$. If the answer is no, mark the region γ *good* and mark all the chains in P'_γ *good*; otherwise, mark the region γ *bad* and mark all the chains in P'_γ *bad*. If γ is good, all chains in P'_γ do not intersect each other and stay inside γ (since chains in P' do not intersect the vertical sides of $\partial\gamma$). Thus, good chains do not intersect each other. On the other hand, the number of bad regions is at most $O(X)$, since we can charge each bad region to a unique intersection (note that each chain is in just one P'_γ); so, the number of bad chains is at most $O(bstX)$, since each P'_γ has at most $O(bs \cdot t)$ chains.

6. For each bad chain p' in P', find all regions γ in the t-division that are intersected by p'. To do so, begin with the region γ containing the start vertex of p'. Find the next line segment $e \in Q_\gamma$ intersected by p', if exists, by brute force in $O(\ell t)$ time, since p' has size at most $O(\ell)$ and Q_γ has size at most $O(t)$. Then reset γ to the neighboring region defined by e, and repeat. The total time is at most $O(\ell t \cdot bstX)$, since every region γ visited by p' after the first can be charged to a unique intersection, and there are at most $O(bstX)$ bad chains p'.

7. For each bad chain p' in P' and each good region γ intersected by p' (all such regions have been identified from step 6), compute all intersections of p' with the good chains of P'_γ by brute force in $(\ell \cdot \ell bst)$ time (since P'_γ has at most $O(\ell \cdot bst)$ edges). The total time is at most $O(\ell \cdot \ell bst \cdot bstX)$.

8. Lastly, compute all intersections among the bad chains, which have $n'' = O(\ell \cdot bstX)$ edges, e.g., by the Bentley–Ottmann sweep [4] in $O((n''+X)\log n'')$ time.

The total running time for Step 1 is $O(n + h\log h)$. The running time for Step 3 is $O(|R|\log|R| + X) = O((n/\sqrt{s}+h)\log n + X)$ time. The total running time for Steps 4–5 is at most $O(n + ((|R|+X)/\sqrt{t}) \cdot bs \cdot \log n + h\log n)$. The total running time for Steps 6–8 is $O((b\ell st)^{O(1)}X)$.

Setting $s = \log^2 n$ and $t = b^2\log^6 n$ gives an overall time bound of $O(n + h\log h + (b\ell)^{O(1)}X\log^{O(1)} n)$.

By initially subdividing the input chains into subchains of length $\log n$, we can lower ℓ to $\log n$ while increasing h to $h + O(n/\log n)$; the $O(h\log h)$ term would then be increased to $O(n + h\log h)$.

After computing all intersections, we can turn P into a collection of at most h disjoint simple chains with $O(n+X)$ vertices, by making tiny cuts to eliminate cycles and taking an Euler tour of each connected component (to traverse the components, we may need to sort intersection points along each edge, but this costs at most $O(X\log n)$ time). Afterwards, we can apply Theorem 1 to compute $TD(P)$ in $O(n + X + h\log h)$ additional time. ◀

Finally, we combine the above lemma with an $n^{\Theta(\varepsilon)}$-way divide-and-conquer to solve the general problem:

▶ **Theorem 3.** *Given a collection P of h (not necessarily simple) polygonal chains with a total of n vertices, we can compute $TD(P)$ in $O(n + h \log h + Xn^{\varepsilon})$ deterministic time for an arbitrarily small constant $\varepsilon > 0$, where X is the number of intersections.*

Proof. Divide P into b subcollections $P_1, \ldots, P_b$, each with n/b vertices, possibly by splitting at most b of the input chains. Suppose P_i has h_i chains and X_i intersections. Note that $\sum_i h_i \leq h + b$ and $\sum_i X_i \leq X$. Recursively compute $TD(P_i)$, whose output gives us the X_i intersections of each P_i. Subsequently, P_i can be turned into a non-intersecting collection of at most h_i simple chains with $O(n/b + X_i)$ vertices. Now, apply Lemma 2 to compute $TD(P)$ in $O(n + X + h \log h + b^{O(1)} X \log^{O(1)} n)$ time.

The total running time satisfies the recurrence

$$T(n, h, X) \leq \max_{\substack{h_1, \ldots, h_b, \\ X_1, \ldots, X_b : \\ \sum_i h_i \leq h + b, \\ \sum_i X_i \leq X}} \left(\sum_{i=1}^{b} T(n/b, h_i, X_i) + O(n + h \log h + b^{O(1)} X \log^{O(1)} n) \right),$$

which solves to $T(n, h, X) = O((n + h \log h + b^{O(1)} X \log^{O(1)} n) \cdot \log_b n)$. Setting $b = n^{\varepsilon/c}$ for a sufficiently large constant c yields the result. ◀

The main open question is whether the running time of the above theorem could be improved to $O(n + h \log h + X)$. A more modest goal is to find a deterministic algorithm with running time $O(n + h \log h + X \log^{O(1)} n)$.

References

1. Nancy M. Amato, Michael T. Goodrich, and Edgar A. Ramos. A randomized algorithm for triangulating a simple polygon in linear time. *Discret. Comput. Geom.*, 26(2):245–265, 2001. doi:10.1007/S00454-001-0027-X.
2. Ivan J. Balaban. An optimal algorithm for finding segments intersections. In *Proc. 11th Annual Symposium on Computational Geometry (SoCG)*, pages 211–219, 1995. doi:10.1145/220279.220302.
3. Reuven Bar-Yehuda and Bernard Chazelle. Triangulating disjoint Jordan chains. *Int. J. Comput. Geom. Appl.*, 4(4):475–481, 1994. doi:10.1142/S0218195994000252.
4. Jon Louis Bentley and Thomas Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Computers*, 28(9):643–647, 1979. doi:10.1109/TC.1979.1675432.
5. Bernard Chazelle. Triangulating a simple polygon in linear time. *Discret. Comput. Geom.*, 6:485–524, 1991. doi:10.1007/BF02574703.
6. Bernard Chazelle and Herbert Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39(1):1–54, 1992. doi:10.1145/147508.147511.
7. Danny Z. Chen and Haitao Wang. A new algorithm for computing visibility graphs of polygonal obstacles in the plane. *J. Comput. Geom.*, 6(1):316–345, 2015. doi:10.20382/JOCG.V6I1A14.
8. Kenneth L. Clarkson, Richard Cole, and Robert Endre Tarjan. Randomized parallel algorithms for trapezoidal diagrams. *Int. J. Comput. Geom. Appl.*, 2(2):117–133, 1992. doi:10.1142/S0218195992000081.
9. Kenneth L. Clarkson and Peter W. Shor. Application of random sampling in computational geometry, II. *Discret. Comput. Geom.*, 4:387–421, 1989. doi:10.1007/BF02187740.
10. Kenneth L. Clarkson, Robert Endre Tarjan, and Christopher J. Van Wyk. A fast Las Vegas algorithm for triangulating a simple polygon. *Discret. Comput. Geom.*, 4:423–432, 1989. doi:10.1007/BF02187741.

11 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008. doi:10.1007/978-3-540-77974-2.

12 David Eppstein and Michael T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *Proc. 16th ACM International Symposium on Advances in Geographic Information Systems (GIS)*, page 16, 2008. doi:10.1145/1463434.1463455.

13 David Eppstein, Michael T. Goodrich, and Darren Strash. Linear-time algorithms for geometric graphs with sublinearly many edge crossings. *SIAM J. Comput.*, 39(8):3814–3829, 2010. doi:10.1137/090759112.

14 Alain Fournier and Delfin Y. Montuno. Triangulating simple polygons and equivalent problems. *ACM Trans. Graph.*, 3(2):153–174, 1984. doi:10.1145/357337.357341.

15 Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987. doi:10.1137/0216064.

16 Michael T. Goodrich. Planar separators and parallel polygon triangulation. *J. Comput. Syst. Sci.*, 51(3):374–389, 1995. doi:10.1006/JCSS.1995.1076.

17 David Haussler and Emo Welzl. ε-nets and simplex range queries. *Discret. Comput. Geom.*, 2:127–151, 1987. doi:10.1007/BF02187876.

18 Kurt Hoffman, Kurt Mehlhorn, Pierre Rosenstiehl, and Robert Endre Tarjan. Sorting Jordan sequences in linear time using level-linked search trees. *Inf. Control.*, 68(1-3):170–184, 1986. doi:10.1016/S0019-9958(86)80033-X.

19 Sanjiv Kapoor and S. N. Maheshwari. Efficiently constructing the visibility graph of a simple polygon with obstacles. *SIAM J. Comput.*, 30(3):847–871, 2000. doi:10.1137/S0097539795253591.

20 David G. Kirkpatrick, Maria M. Klawe, and Robert Endre Tarjan. Polygon triangulation in $O(n \log\log n)$ time with simple data structures. *Discret. Comput. Geom.*, 7:329–346, 1992. doi:10.1007/BF02187846.

21 Philip N. Klein, Shay Mozes, and Christian Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Proc. 45th ACM Symposium on Theory of Computing (STOC)*, pages 505–514, 2013. doi:10.1145/2488608.2488672.

22 Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980. doi:10.1137/0209046.

23 Jirí Matoušek. Derandomization in computational geometry. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 559–595. North Holland / Elsevier, 2000. doi:10.1016/B978-044482537-7/50014-0.

24 Joseph S. B. Mitchell, Valentin Polishchuk, Mikko Sysikaski, and Haitao Wang. An optimal algorithm for minimum-link rectilinear paths in triangulated rectilinear domains. *Algorithmica*, 81(1):289–316, 2019. doi:10.1007/S00453-018-0446-1.

25 Ketan Mulmuley. A fast planar partition algorithm, I. *J. Symb. Comput.*, 10(3/4):253–280, 1990. doi:10.1016/S0747-7171(08)80064-8.

26 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer, 1985. doi:10.1007/978-1-4612-1098-6.

27 Raimund Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom.*, 1:51–64, 1991. doi:10.1016/0925-7721(91)90012-4.

28 Robert Endre Tarjan and Christopher J. Van Wyk. An $O(n \log\log n)$-time algorithm for triangulating a simple polygon. *SIAM J. Comput.*, 17(1):143–178, 1988. doi:10.1137/0217010.

29 Haitao Wang. A new algorithm for Euclidean shortest paths in the plane. *J. ACM*, 70(2):11:1–11:62, 2023. doi:10.1145/3580475.

A When randomization is allowed

In this appendix, we describe a simpler randomized algorithm, which can be viewed as a variant of an algorithm for $h = 1$ by Eppstein, Goodrich, and Strash [13]:

▶ **Theorem 4.** *Given a collection P of h (not necessarily simple) polygonal chains with a total of n vertices, we can compute $TD(P)$ in $O(n + h \log h + X \log^{(c)} n)$ expected time for an arbitrarily large constant c, where X is the number of intersections.*

Proof. Let E be the edge set of P. Let s and t be parameters to be set later. The algorithm works as follows:

1. Let R be a random subset of E of size n/s.

2. Compute $TD(R)$ in $O(|R| \log |R| + X)$ time [6, 9, 25], and compute a t-division of $TD(R)$. Preprocess $TD(R)$ for point location [26] in $O(|R|)$ time.

3. For each trapezoid $\tau \in TD(R)$, we generate the list E_τ of all edges of E intersecting τ (called a "conflict list"). To do this, we begin with the trapezoid τ containing the start vertex of a chain, which can be found in $O(\log n)$ time by point location. Let e be the first edge of the chain. We then walk along e in $TD(R)$ to identify the next trapezoid intersected by e; this requires $O(\deg(\tau))$ time, where $\deg(\tau)$ denotes the number of neighboring trapezoids of τ. We continue until we encounter the endpoint of e, then reset e to the next edge, and repeat. The total time is $O(h \log n)$ for the h point location queries, plus $O(\sum_{\tau \in TD(R)} |E_\tau| \cdot \deg(\tau))$ for the walks.

   Clarkson and Shor [9, Lemma 4.3 and Corollary 4.4] showed that

   (i) $\sum_{\tau \in TD(R)} |E_\tau| \cdot \deg(\tau)$ has expected value $O((n/s + X/s^2) \cdot s) = O(n + X/s)$;
   (ii) $\max_{\tau \in TD(R)} |E_\tau| = O(s \log n)$ with probability at least (say) $1 - O(1/n^2)$.

   If (ii) fails, we can afford to switch to a naive $O(n^2)$-time algorithm. So, from now on, assume that (ii) holds.

4. For each region γ in the t-division and each (vertical or non-vertical) side e of the boundary $\partial\gamma$, find the intersections of e with P. There are $O(s \log n)$ such intersections by (ii) and they can be generated in $O(s \log n)$ time (from the output of Step 3). By summing over all $O(|R|/\sqrt{t})$ choices of e, there are $O((|R|/\sqrt{t}) \cdot s \log n)$ such intersections in total and they can be generated in $O((|R|/\sqrt{t}) \cdot s \log n)$ time.

   Cut the chains of P at all these intersection points to obtain a new collection P' of polygonal chains. Note that the intersections are generated in sorted order along the chains from Step 3. The number of vertices in P' is $n' \le n + O((|R|/\sqrt{t}) \cdot s \log n)$, and the number of chains in P' is $h' \le h + O((|R|/\sqrt{t}) \cdot s \log n)$.

5. For each region γ in the t-division, let P'_γ be the chains of P' inside γ, and compute $TD(P'_\gamma)$ recursively. Let n'_γ and h'_γ be the number of vertices and chains in P'_γ respectively. Note that $\sum_\gamma n'_\gamma = n'$ and $\sum_\gamma h'_\gamma = h'$. Furthermore, $\max_\gamma n'_\gamma \le O(t \cdot s \log n)$ by (ii).

   Afterwards, we can piece together $TD(P'_\gamma)$ (after clipping to γ) over all regions γ, to obtain $TD(P)$ in $O(n')$ additional time.

Set $s = \log n$ and $t = \log^6 n$. The total expected running time satisfies the recurrence

$$T(n,h,X) \;\leq \max_{\substack{\sum_\gamma n'_\gamma \leq n + O(n/\log n) \\ \sum_\gamma h'_\gamma \leq h + O(n/\log n) \\ \sum_\gamma X_\gamma \leq X \\ \max_\gamma n'_\gamma \leq O(\log^8 n)}} \left(\sum_\gamma T(n'_\gamma, h'_\gamma, X_\gamma) + O(n + h\log n + X) \right).$$

For the base case, we have $T(n,h,0) = O(n + h\log^{O(1)} n)$ by Bar-Yehuda and Chazelle's algorithm [3], and $T(n,h,X) = O(n^2)$ by a naive algorithm for $X > 0$. Applying the recursion for $c+1$ levels before switching to the base case yields a total time bound of $O(n + h\log n + \sum_{j=1}^{c+1}(h + n/\log^{(j)} n)\log^{O(1)}(\log^{(j)} n) + X((\log^{(c+1)} n)^8)^2) \leq O(n + h\log h + X\log^{(c)} n)$. ◀

Charting the Diameter Computation Landscape of Intersection Graphs in 3D and Above

Timothy M. Chan
Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign, IL, USA

Hsien-Chih Chang
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Jie Gao
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Sándor Kisfaludi-Bak
Department of Computer Science, Aalto University, Espoo, Finland

Hung Le
Manning CICS, University of Massachusetts Amherst, Amherst, MA, USA

Da Wei Zheng
Institute of Science and Technology Austria, Klosterneuburg, Austria

Abstract

Recent research on computing the diameter of geometric intersection graphs has made significant strides, primarily focusing on the 2D case [14, 11, 8] where truly subquadratic-time algorithms were given for simple objects such as unit-disks and (axis-aligned) squares. However, in three or higher dimensions, there is no known truly subquadratic-time algorithm for any *intersection graph of* non-trivial objects, even basic ones such as unit balls or (axis-aligned) unit cubes. This was partially explained by the pioneering work of Bringmann et al. [5] which gave several truly subquadratic lower bounds, notably for unit balls or unit cubes in 3D when the graph diameter Δ is at least $\Omega(\log n)$, hinting at a pessimistic outlook for the complexity of the diameter problem in higher dimensions. In this paper, we substantially extend the landscape of diameter computation for objects in three and higher dimensions, giving a few positive results. Our highlighted findings include:

1. A truly subquadratic-time algorithm for deciding if the diameter of unit cubes in 3D is at most 3 (Diameter-3 hereafter), the first algorithm of its kind for objects in 3D or higher dimensions. Our algorithm is based on a novel connection to pseudolines, which is of independent interest.
2. A truly subquadratic time lower bound for Diameter-3 of unit balls in 3D under the Orthogonal Vector (OV) hypothesis, giving the first separation between unit balls and unit cubes in the small diameter regime. Previously, computing the diameter for both objects was known to be quadratic hard when the diameter is $\Omega(\log n)$ [5].
3. A near-linear-time algorithm for Diameter-2 of unit cubes in 3D, generalizing the previous result for unit squares in 2D [5].
4. A truly subquadratic-time algorithm and lower bound for Diameter-2 and Diameter-3 of rectangular boxes (of arbitrary dimension and sizes), respectively.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Graph Diameter, Geometric Intersection Graphs, Unit Ball Graphs

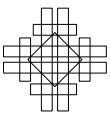
Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.29

Related Version *Full Version*: https://arxiv.org/abs/2603.21790

Funding *Timothy M. Chan*: Supported by NSF grant CCF-2224271.
Hsien-Chih Chang: Supported by NSF CAREER award CCF-2443017.
Jie Gao: Supported by NSF DMS-2220271, DMS-2311064, IIS-2229876, CCF-2118953, CNS-2515159.
Sándor Kisfaludi-Bak: Supported by the Research Council of Finland, Grant 363444.
Hung Le: Supported by an NSF grant CCF-2517033 and an NSF CAREER Award CCF-2237288.



© Timothy M. Chan, Hsien-Chih Chang, Jie Gao, Sándor Kisfaludi-Bak, Hung Le, and Da Wei Zheng;
licensed under Creative Commons License CC-BY 4.0



42nd International Symposium on Computational Geometry (SoCG 2026).
Editors: Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri; Article No. 29; pp. 29:1–29:15



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Da Wei Zheng: This project has received funding from the Austrian Science Fund (FWF) grant DOI 10.55776/I5982. For open access purposes, the author has applied a CC BY public copyright license to any author-accepted manuscript version arising from this submission.

1 Introduction

Computing the diameter of sparse graphs is *quadratic hard*: under the Strong Exponential Time Hypothesis (SETH), there is no $O(n^{2-\varepsilon})$ algorithm for distinguishing diameter 2 vs. 3 of graphs with n vertices and $\tilde{O}(n)$[1] edges [19]. That is, the trivial algorithm for computing the diameter by doing BFS from every vertex is essentially optimal. Since then, the major research focus has been on substantially beating the BFS-based algorithm for structural families of graphs, especially planar/minor-free graphs and geometric intersection graphs. For planar/minor-free graphs, truly-subquadratic algorithms were known [6, 15, 13, 16, 8]. For geometric intersection graphs, the complexity of computing the diameter remains poorly understood, due to the sheer diversity of geometric objects underlying the graphs and the fact that geometric intersection graphs can have $\Omega(n^2)$ edges.

The pioneering work of Bringmann *et al.* [5] studied the diameter of intersection graphs of several types of objects. In dimension three and above, they considered (axis-parallel) hypercubes[2] and balls. Let Diameter-Δ be the problem to decide if a given input graph has diameter at most Δ or at least $\Delta+1$. They showed that it is quadratic hard to solve: (1) Diameter-2 for hypercubes in $\mathbb{R}^{12}$ under the (3-uniform 6-)Hyperclique Hypothesis; (2) Diameter-3 for (not necessarily axis-aligned) unit segments and equilateral triangles in $\mathbb{R}^2$ under the Orthogonal Vector (OV) Hypothesis; and (3) Diameter-$\Omega(\log n)$ for unit balls and unit cubes in $\mathbb{R}^3$, and axis-parallel line segments in $\mathbb{R}^2$, all under the OV Hypothesis.

The hardness results can be interpreted as indicating that the radius-Δ neighborhood balls of each geometric shape class are sufficiently complex and expressive, capable of encoding the hard instances used in various fine-grained reductions. One possible way to quantify the complexity of the set system of neighborhood balls is the notion of *VC-dimension*. Given a set system $(U, \mathcal{F})$ with a ground set U and a family $\mathcal{F}$ of subsets of U, its *VC-dimension* is the cardinality of the largest $S \subseteq U$ such that S is *shattered* by $\mathcal{F}$ – for every $S' \subseteq S$, there is some $X \in \mathcal{F}$ such that $X \cap S = S'$. We say that a graph G has *distance VC-dimension* at most d if the set system of radius-r neighborhood balls $(V_G, \{N^r[v]\}_{r\geq 0})$ has VC dimension at most d. Planar graphs (more generally, minor-free graphs) [12, 4, 17, 13, 16] and intersection graphs of pseudo-disks [1, 14, 11] both have bounded distance VC-dimension, whereas intersection graphs of unit segments and equilateral triangles do not. Recently Chan *et al.* [8] gave truly-subquadratic-time diameter algorithms for squares and unit-disks in the plane – both shapes being pseudo-disks, thus with finite distance VC-dimension. Together, these results hinted at *finite VC-dimension* as an overarching property for fast diameter computation.

We undertake a comprehensive study of computing diameter on geometric intersection graphs. Our results are presented in a pair of papers. While the companion paper [9] focuses solely on the 2D case, this paper is devoted to three or higher dimensions. In the higher-dimensional case, there is no known truly-subquadratic-time algorithm for any non-trivial type of objects, even for basic ones such as unit balls or unit cubes. Upper bound techniques for the 2D cases [5, 14, 11, 8] heavily rely on planarity in various places, most notably in bounding the VC dimension using the non-planarity of K_5 [14, 11, 8]. These planarity-specific techniques, in addition to the negative results by Bringmann *et al.* [5], hint at a pessimistic outlook for the complexity of diameter problems in higher dimensions.

[1] $\tilde{O}(\cdot)$ hides polylogarithmic factors in n.
[2] In this paper, hypercubes and boxes are axis-parallel by default, unless noted otherwise.

Table 1 Previous and new time bounds for deciding whether an intersection graph of geometric objects has diameter at most Δ. All hypercubes/boxes are axis-aligned. New results are shown in bold. Conditional lower bounds marked "(OV)", "(3H6)", and "(combK4)" assume the Orthogonal Vectors hypothesis, the 3-uniform 6-hyperclique hypothesis, and the combinatorial 4-clique hypothesis, respectively, where the latter is for combinatorial algorithms (all upper bounds are obtained from combinatorial algorithms). See the full version for definitions. "2D Paper" refers to the companion paper in the pair [9].

Graph class		Lower bound	Upper bound
Unit balls	2D		$O^*(n^{2-1/18})$ for general Δ [8] $O^*(n^{2-1/9})$ for $\Delta = O(1)$ [8] $\tilde{O}(n^{4/3})$ for $\Delta = 2$ (2D Paper [9])
	3D	$\Omega^*(n^2)$ for general Δ (OV) [5] $\Omega^*(n^2)$ for $\Delta = 3$ (OV) (Thm. 9)	
	dD	$\Omega^*(n^2)$ for $\Delta = 2$ & $d = 7$ (combK4) (Thm. 5)	
Unit hypercubes	2D		$O(n \log n)$ for $\Delta = 2$ [5] $O^*(n^{7/4})$ for $\Delta = O^*(1)$ [14] $\tilde{O}(n^{2-1/8})$ for general Δ [8] $O^*(n)$ for const. Δ (2D Paper [9])
	3D	$\Omega^*(n^2)$ for general Δ (OV) [5]	$\tilde{O}(n)$ for $\Delta = 2$ (Thm. 3) $\tilde{O}(n^{2-1/13})$ for $\Delta = 3$ (Thm. 3)
	4D	$\Omega^*(n^2)$ for $\Delta = 3$ (OV) (Thm. 5)	
	dD	$\Omega^*(n^2)$ for $\Delta = 2$ & $d = 12$ (3H6) [5] $\Omega^*(n^2)$ for $\Delta = 2$ & $d = 10$ (3H6) (Thm. 5) $\Omega^*(n^2)$ for $\Delta = 2$ & $d = 6$ (combK4) (Thm. 5)	
Cubes	2D		$O^*(n^{2-1/12})$ for general Δ [8]
	3D	$\Omega^*(n^2)$ for general Δ (OV) [5] $\Omega^*(n^2)$ for $\Delta = 3$ (OV) (Thm. 3)	$\tilde{O}(n^{2-1/5})$ for $\Delta = 2$ (Thm. 7)
Boxes	2D	$\Omega^*(n^2)$ for $\Delta = 3$ (OV) (Thm. 3)	$\tilde{O}(n^{7/4})$ for $\Delta = 2$ (Full ver.)
	3D		$\tilde{O}(n^{2-1/6})$ for $\Delta = 2$ (Thm. 7)

▶ **Question 1.** *Does a truly-subquadratic-time algorithm for computing the diameter exist for any natural class of geometric intersection graphs in 3D or higher dimensions?*

As a starting point, the hardness landscape for unit cubes seems peculiar. Recall that Bringmann *et al.* [5] showed that in 12D, distinguishing diameter 2 versus 3 is already quadratic hard (under the Hyperclique Hypothesis). However in 3D, the quadratic hardness does not kick in until $\Delta \geq \Omega(\log n)$. Duraj, Konieczny, and Potępa [14] solves the 2D case in time $\tilde{O}(\Delta n^{7/4})$ for Diameter-Δ, which was later extended by [8] to solve Diameter in $\tilde{O}(n^{2-1/8})$ time for general Δ. What about unit cubes in 3D? The same question can be asked about unit balls, which typically behave similarly to unit cubes.

▶ **Question 2.** *What is the complexity landscape of Diameter-Δ for unit cubes and unit balls in constant dimensions? Do radius-Δ neighborhood balls of their intersection graphs have bounded VC-dimension, in particular in 3D?*

1.1 Our Contributions

We provide a comprehensive study on basic objects in higher dimensions; our results are summarized in Table 1. As corollaries, we make significant progress towards Question 1 and Question 2. Below, we discuss each of the contributions in more detail.

Unit cubes versus unit balls. Our first result is a separation between unit (hyper)cubes and unit balls: Diameter-3 for unit balls is hard, whereas Diameter-3 for unit cubes admits a truly subquadratic-time algorithm.

▶ **Theorem 3.** *There is no $O(n^{2-\varepsilon})$ algorithm for Diameter-3 for unit balls in $\mathbb{R}^3$ under the OV hypothesis for any $\varepsilon > 0$, while for unit cubes in $\mathbb{R}^d$ for $d \geq 3$, Diameter-3 and Diameter-2 can be solved in $\tilde{O}(n^{2-1/13})$ and $\tilde{O}(n)$ time, respectively.*

Our lower bound for Diameter-3 of unit balls is by reducing from Diameter-2 of sparse tripartite graphs $G = (A \cup B \cup C, E)$, which is quadratic hard under the OV hypothesis. Our first idea for the reduction would be to create a unit ball for each vertex in $A \cup B \cup C$, as well as a unit ball for each edge in E. We encode edges in the original graph G by intersection: a vertex $u \in A \cup B \cup C$ is incident to an edge $uv \in E$ if and only if the unit balls corresponding to u and uv intersect. Then a length-2 path $a \to b \to c$ in G will correspond to a length-4 path $(\!|a|\!) \to (\!|ab|\!) \to (\!|b|\!) \to (\!|bc|\!) \to (\!|c|\!)$ in the intersection graph, where $(\!|x|\!)$ is the unit ball corresponding to (a vertex or an edge) x. With this idea, we could obtain hardness of Diameter-4. To obtain a lower bound for Diameter-3, we do not create unit balls for the middle set of vertices B; the path would then be $(\!|a|\!) \to (\!|ab|\!) \to (\!|bc|\!) \to (\!|c|\!)$. The difficulty is to guarantee that $(\!|ab|\!)$ only intersects $(\!|bc|\!)$, not $(\!|\tilde{b}c|\!)$ for some other vertex $\tilde{b} \in B$; in the Diameter-4 case, this is enforced by the ball $(\!|b|\!)$. Here, we achieve the guarantee by introducing an angle parameter δ to control the intersections of the balls and prevent unwanted edges from being added. See Section 2.

The algorithms for Diameter-2 and 3 for unit cubes in Theorem 3 are perhaps most interesting. First, we show that *the VC-dimension of the 2- and 3-neighborhoods of 3D unit cubes is bounded by a constant.* Here we depart from existing techniques for the 2D case [14, 11, 8], which rely on the non-planarity of K_5. In a small region, unit cubes behave like orthants, and so the intersection between unit cubes corresponds to a *dominance* relation. We divide the sequence of dominance relations on the length-2 and length-3 path into a number of special cases, bound the VC-dimension of each by 1 or 2 via direct arguments, and then bound the overall VC-dimension by combining subsystems together. See Section 3.1 and Section 4.1 for details.

Our VC-dimension bound, in combination with existing techniques [8], is enough to imply truly-subquadratic-time algorithms for Diameter-2 and Diameter-3, but because of large constants, the running time would be awfully close to quadratic (about $O(n^{1.99999992})$ for Diameter-3). We further build on the proof ideas to give ultra efficient algorithms: In the case of Diameter-2, because the subsystems from various cases actually all have VC-dimension 1, we are able to use orthogonal range searching, in combination with divide-and-conquer, to achieve *near-linear time.* (Large constants appear in the logarithmic factors, instead of in the main polynomial factor.) For Diameter-3, because the subsystems from various cases actually turn out to be special kinds of 2-dimensional systems corresponding to *pseudoline arrangements*, we are able to use known range searching techniques, in combination with a grid approach, to achieve a more reasonable running time $\tilde{O}(n^{2-1/13})$. (We find it surprising that pseudolines turn out to be relevant for a problem about 3D unit cubes!) The technical details are intricate, as we need to work with abstract pseudolines that are only implicitly represented via certain oracles. See Section 3 and §4 of full paper for details.

Our constant bound on the VC-dimension of the 2- and 3-neighborhoods of 3D unit cube graphs suggests an intriguing open question of independent interest to combinatorial geometers: do r-neighborhoods similarly have bounded VC-dimension for all constant $r > 3$?

▶ **Conjecture 4** (VC-Dimension of Unit-Cube Graphs). *There exists a function f such that the VC dimension of $(V, \{N^r[v]\}_{v\in V, r\geq 0})$ is at most $f(r)$, where $N^r[v]$ is the r-neighborhood of v in the unit cube graph $G = (V, E)$.*

Combining with [8], Conjecture 4 would imply that unit-cube graphs of constant diameter admit a truly-subquadratic-time algorithm for computing diameter, which is a natural class of intersection graphs motivated by Question 1.

Unit-hypercubes in 4^+D. Next, we show that Diameter-3 is quadratic hard for unit hypercubes in 4D. Therefore, there is no longer a meaningful separation between unit hypercubes and unit balls in dimension 4. When the dimension is even higher, say at least 6D, we can get hardness results for Diameter-2, under the *3-uniform 6-hyperclique (3H6)* hypothesis or *combinatorial 4-clique (combK4)* hypothesis, by refining Bringmann *et al.*'s previous proof [5] which required 12 dimensions. We also rule out near-linear time combinatorial algorithms in 4D under the *combinatorial boolean matrix multiplication (BMM)* hypothesis. See Section 2 of full paper for details.

▶ **Theorem 5.** *Let $\varepsilon \in (0, 1)$ be any given parameter. There is no $O(n^{2-\varepsilon})$ algorithm for:*
1. *Diameter-3 for unit hypercubes in $\mathbb{R}^4$ under the OV hypothesis.*
2. *Diameter-2 of unit balls in $\mathbb{R}^7$ under the CombK4 hypothesis.*
3. *Diameter-2 for unit hypercubes in $\mathbb{R}^6$ and $\mathbb{R}^{10}$ under CombK4 / 3H6 hypotheses, resp.*

▶ **Theorem 6.** *Let $\varepsilon \in (0, 1)$ be any given parameter. There is no combinatorial $O(n^{3/2-\varepsilon})$ algorithm for Diameter-2 of unit hypercubes in $\mathbb{R}^4$ under the combinatorial BMM hypothesis.*

Cubes and boxes. In 2D, the intersection graphs of (non-unit) squares admit a truly subquadratic-time algorithm for computing the diameter (of any value) [8]. In 3D, we show that for (non-unit) cubes, Diameter-3 is quadratic-hard; this, along with the algorithms in Theorem 3, gives a separation between cubes and their unit counterparts. In contrast, we show that Diameter-2 of (non-unit) cubes is solvable in truly-subquadratic time. Even for the more general case of 3D boxes, we also obtain a truly-subquadratic-time algorithm. (Note that this result is new even for 2D rectangles!) Instead of relying on VC-dimension, this subquadratic algorithm uses a different, grid-based approach. Grid-based approaches have been used before for solving other problems about boxes [7, 10], but one challenge arises from the fact that 3D boxes (or cubes) have quadratic *union complexity*, unlike 3D unit cubes or orthants. We show that in the case of diameter 2, some of the sides are irrelevant and some boxes can actually be replaced by orthants. See Section 5 of full paper for details.

▶ **Theorem 7.** *There is no $O(n^{2-\varepsilon})$ algorithm for Diameter-3 for (non-unit) cubes in $\mathbb{R}^3$ under the OV hypothesis for any $\varepsilon > 0$. On the other hand, Diameter-2 can be solved in time $\tilde{O}(n^{2-1/5})$ for cubes and $\tilde{O}(n^{2-1/6})$ for boxes in $\mathbb{R}^3$.*

Finite VC-dimension as a sufficient condition? Our collection of results for many types of objects in 2D (in the companion paper [9]) and higher-dimensional cases reinforces the same pattern: truly subquadratic-time algorithms are mostly for graphs with bounded VC dimension. We conjecture that these are special cases of a broader phenomenon.

▶ **Conjecture 8** (VC-Dimension vs Diameter Conjecture). *Diameter-Δ of the intersection graph of low-complexity geometric objects (regardless of the dimension) can be solved in truly-subquadratic time if the VC dimension of Δ-neighborhoods is bounded by a constant.*

Here, by low-complexity, we mean that the geometric object has constant description complexity. A positive resolution of Conjecture 8 would provide a powerful algorithmic tool for solving the diameter problem on geometric intersection graphs. A non-trivial test case for Conjecture 8 is (non-unit) disks in 2D: the distance VC dimension is at most 4 [11] whereas currently there is no known truly-subquadratic algorithm even for DIAMETER-2.

It is tempting to strengthen Conjecture 8 to an "if and only if" statement; however, our truly-subquadratic algorithm for DIAMETER-2 of boxes in Theorem 7 provides a counter-example: the intersection graphs of diameter 2 for boxes and rectangles have unbounded VC-dimension. (For proof of unbounded VC-dimension, see the full paper for details)

Graph preliminaries. Throughout this paper, let $G = (V, E)$ be a geometric intersection graph on n geometric objects $\mathcal{O}$. The input will be $\mathcal{O}$ and the graph will be implicit. We use $d(u, v)$ to denote the distance between two vertices $u, v \in V$ in G. We denote the 1-hop *neighborhood* of a vertex $v \in V$ by $N[v]$, and the *r-neighborhood ball* of v by $N^r[v] := \{u \in V : d(u, v) \leq r\}$.

Shatter dimension. For a set system $(U, \mathcal{F})$, define $Sh(n) := \max_{|S|=n} |\{S \cap F : F \in \mathcal{F}\}|$. The *shatter dimension* of the set system is the minimum d such that $Sh(n) = O(n^d)$. By the Sauer-Shelah Lemma [20, 21], if the VC-dimension is d, then the shatter dimension is d. The converse partially holds: if the shatter dimension is d, then the VC-dimension is $O(d \log d)$.

2 Diameter-3 for Unit Balls in 3D

▶ **Theorem 9.** *Assuming the OV hypothesis, there is no $O(n^{2-\varepsilon})$ time algorithm for deciding if the intersection graph of a given set of n unit balls in $\mathbb{R}^3$ has diameter at most 3.*

The reduction is from DIAMETER-2 for n-vertex *sparse tripartite graphs* $G = (A \cup B \cup C, E)$, where the number of edges m is $\tilde{O}(n)$. It is well-known that solving DIAMETER-2 for sparse tripartite graphs in truly-subquadratic time would break the OV hypothesis [19].

▶ **Lemma 10** (DIAMETER-2 sparse tripartite graphs)**.** *There is no $O(n^{2-\varepsilon})$ time algorithm for deciding if a sparse tripartite graph has diameter at most 2, assuming the OV hypothesis.*

In this section, we work with balls of radius 1/2 instead of 1. Let $\varepsilon = \Theta(1/n^4)$ and $\delta = \Theta(1/n^3)$. First, we map every vertex v in $A \cup C$ to a distinct number, also denoted by v, in $[\varepsilon, 2\varepsilon]$, such that $\min_{a_1, a_2 \in A:\, a_1 \neq a_2} |a_1 - a_2| \geq \Omega(\varepsilon/n)$ and $\min_{c_1, c_2 \in C:\, c_1 \neq c_2} |c_1 - c_2| \geq \Omega(\varepsilon/n)$. Vertices of B are mapped to distinct numbers in $[0.9, 1]$ such that $\min_{b_1, b_2 \in B:\, b_1 \neq b_2} |b_1 - b_2| \geq \Omega(1/n)$. Then we create four sets of balls as follows.

1. For every vertex $a \in A$, we add a ball centered at $s_a = (1 + a\cos\delta, a\sin\delta, 0)$.
2. For every edge $(a, b) \in (A \times B) \cap E$, we add a ball centered at $p_{ab} = (1 + a\cos\delta - b\sin\delta, a\sin\delta + b\cos\delta, \sqrt{1 - b^2})$.
3. For every edge $(b, c) \in (B \times C) \cap E$, we add a ball centered at $q_{bc} = (-c, b, \sqrt{1 - b^2})$.
4. For every vertex $c \in C$, we add a ball centered at $t_c = (-c, 0, 0)$.

Let $\mathcal{B}_i$ be the set of balls created in step i for $i \in [4]$ and $\mathcal{B} := \cup_i \mathcal{B}_i$. Observe that:

▶ **Observation 11.** *Let $(\!|p|\!)$ be the ball of radius 1/2 centered at a point p. We have:*

1. *$(\!|s_a|\!) \cap (\!|p_{\tilde{a}b}|\!) \neq \emptyset$ if and only if $a = \tilde{a}$.*
2. *$(\!|p_{ab}|\!) \cap (\!|q_{\tilde{b}c}|\!) \neq \emptyset$ if and only if $b = \tilde{b}$.*
3. *$(\!|p_{bc}|\!) \cap (\!|t_{\tilde{c}}|\!) \neq \emptyset$ if and only if $c = \tilde{c}$.*
4. *Every two balls in $\mathcal{B}_i$ intersect for every $i \in [4]$.*
5. *For any two balls $b_1 \in \mathcal{B}_i$ and $b_2 \in \mathcal{B}_j$ such that $|i - j| \geq 2$, $b_1 \cap b_2 = \emptyset$.*

Proof. For item 1, observe that

$$\begin{aligned}\|s_a - p_{\tilde{a}b}\|_2^2 &= ((a-\tilde{a})\cos\delta + b\sin\delta)^2 + ((a-\tilde{a})\sin\delta - b\cos\delta)^2 + 1 - b^2 \\ &= (a-\tilde{a})^2(\cos^2\delta + \sin^2\delta) + b^2(\sin^2\delta + \cos^2\delta) + 1 - b^2 \;=\; (a-\tilde{a})^2 + 1,\end{aligned}$$

which is exactly 1 if $a = \tilde{a}$, and at least $1 + \Omega(1/n^{10})$ if $|a - \tilde{a}| \geq \Omega(\varepsilon/n)$.

For item 2, observe that

$$\begin{aligned}&\|p_{ab} - q_{\tilde{b}c}\|_2^2 \\ =&(1 + a\cos\delta - b\sin\delta + c)^2 + (a\sin\delta + b\cos\delta - \tilde{b})^2 + (\sqrt{1-b^2} - \sqrt{1-\tilde{b}^2})^2 \\ =&(1 + \Theta(\varepsilon) - \Theta(\delta) + \Theta(\varepsilon))^2 + (\Theta(\varepsilon\delta) + b - \Theta(\delta^2) - \tilde{b})^2 + (\sqrt{1-b^2} - \sqrt{1-\tilde{b}^2})^2 \\ =&(1 - \Theta(1/n^3))^2 + (|b - \tilde{b}| \pm \Theta(1/n^6))^2 + (\sqrt{1-b^2} - \sqrt{1-\tilde{b}^2})^2,\end{aligned}$$

which is $1 - \Theta(1/n^3)$ if $b = \tilde{b}$, and at least $1 + \Omega(1/n^2)$ if $|b - \tilde{b}| \geq \Omega(1/n)$.

For item 3, observe that $\|q_{bc} - t_{\tilde{c}}\|_2^2 = (\tilde{c} - c)^2 + 1$, which is exactly 1 if $c = \tilde{c}$, and at least $1 + \Omega(1/n^{10})$ if $|c - \tilde{c}| \geq \Omega(\varepsilon/n)$. Item 4 is easy to check. For item 5, observe that $\|s_a - t_c\|_2 \geq 1 + a\cos\delta + c \geq 1 + \Omega(1/n^4)$, whereas $\|s_a - q_{bc}\|_2$ and $\|p_{ab} - t_c\|_2$ are both $\sqrt{2} \pm o(1)$. ◀

In the real RAM model, balls $\mathcal{B}$ can clearly be constructed in $O(n+m) = \tilde{O}(n)$ time. In the RAM model, it suffices to approximate all coordinates up to $O(1/n^{10})$ additive error, which can be done in $\tilde{O}(1)$ time (we could actually pick "nice" choices of $b \in B$ so that b and $\sqrt{1-b^2}$ are both rational, and also a nice δ so that $\cos\delta$ and $\sin\delta$ are both rational). Thus, the reduction can be done in $\tilde{O}(n)$ time. Theorem 9 follows from the following lemma.

▶ **Lemma 12.** *Let K be the intersection graph of the set of balls $\mathcal{B}$. Then G has diameter at most 2 if and only if K has diameter at most 3.*

Proof. Observe by item 4 of Observation 11 that the balls in the same set $\mathcal{B}_i$ induce a clique in K. By item 5 of Observation 11, there is no edge between two balls in $\mathcal{B}_i$ and $\mathcal{B}_j$ when $|j - i| \geq 2$. Thus, *K has diameter at most 3 if and only if for every ball $[\![s_a]\!] \in \mathcal{B}_1$ and every ball $[\![t_c]\!] \in \mathcal{B}_4$, $d_K(s_a, t_c) \leq 3$.* Here we slightly abuse the notation by using the center of the ball to denote the corresponding vertex in K.

Suppose that G has diameter at most 2. Let $P = (a, b, c)$ be any path of length 2 in G. Then by Observation 11, $(s_a, p_{ab}, q_{bc}, t_c)$ is a path of length 3 in K. Thus, K has diameter at most 3 by the above observation. For the other direction, if K has diameter at most 3, then for any two vertices s_a and t_c (such that $[\![s_a]\!] \in \mathcal{B}_1$ and $[\![t_c]\!] \in \mathcal{B}_4$), a path of length 3 between s_a and t_c in K must be of the form $(s_a, p_{\tilde{a}b}, q_{\tilde{b}c}, t_c)$. Then again by Observation 11, $a = \tilde{a}$ and $b = \tilde{b}$, and therefore, (a, b, c) is a path of length 2 in G. ◀

3 A Near-Linear Diameter-2 Algorithm for 3D Unit Cubes

In this section, we present an $\tilde{O}(n)$-time algorithm for testing whether the diameter of a 3D unit-cube graph is ≤ 2. This shows that the near-linear Diameter-2 result of Bringmann *et al.* [5] for 2D unit squares surprisingly extends to 3D (ignoring logarithmic factors). It also complements our lower bound result in Theorem 6, which shows conditionally that there are no similar, near-linear combinatorial Diameter-2 algorithms for 4D unit hypercubes.

In Section 3.1, we study the combinatorial problem of bounding the VC-dimension of the distance-2 neighborhoods of 3D unit-cube graphs as a warm-up. In the full version, we use ideas from the combinatorial proof to design a near-linear-time divide-and-conquer algorithm.

For a point $p \in \mathbb{R}^3$, denote $[\![p]\!]$ the unit cube centered at p. For simplicity, assume that the input is in general position, e.g., all coordinates are distinct. We will solve the problem in a slightly more general setting for 3 point sets P, Q, R, testing whether all distances between $[\![p]\!]$ and $[\![r]\!]$ for $(p, r) \in P \times R$ are exactly 2 in the tripartite intersection graph of the unit cubes centered at P, Q, R. The original problem reduces to the case when P, Q, R are all equal to the input point set (or almost equal, if we want to ensure general position); the diameter of the original point set is at most 2 iff every pair in $P \times R$ has distance exactly 2.

3.1 VC-dimension bound

For two points $p, q \in \mathbb{R}^3$, write $p \prec_x q$ if p has smaller x-coordinate than q, and $p \succ_x q$ if p has larger x-coordinate than q. Define $\prec_y$, $\succ_y$, $\prec_z$, $\succ_z$ similarly. Define the trivial relation $\top$ which is always true. A *generalized dominance relation* is a relation $\lhd$ where $p \lhd q$ iff $p \lhd_x q$ and $p \lhd_y q$ and $p \lhd_z q$, for some choices of $\lhd_x \in \{\prec_x, \succ_x, \top\}$, $\lhd_y \in \{\prec_y, \succ_y, \top\}$, and $\lhd_z \in \{\prec_z, \succ_z, \top\}$.

The motivation for considering generalized dominance relations is this: Consider a uniform grid with unit side-length. Inside a grid cell, any unit cube is equivalent to an orthant. If p lies in a grid cell and q lies in a neighboring grid cell, then the condition that $[\![p]\!]$ and $[\![q]\!]$ intersect corresponds precisely to $p \lhd q$ for some generalized dominance relation.

Let P, Q, R be 3 point sets in $\mathbb{R}^3$. Let $\lll = (\lhd_1, \lhd_2)$ be 2 generalized dominance relations in $\mathbb{R}^3$. For each $r \in R$, we can write $N^2_{\lll}[r]$ as $\{p \in P : \exists q \in Q \text{ with } p \lhd_1 q \text{ and } q \lhd_2 r\}$. Define the set system $\mathcal{S}_{\lll}(P, Q, R) := (P, \{N^2_{\lll}[r] : r \in R\})$. We first prove that this set system has bounded VC-dimension by bounding its shatter dimension.

Previous proofs of bounded distance VC-dimension usually involve planarity arguments (avoidance of K_5), but for the 3D problem here, we use a different strategy. We divide into easier cases based on the separability of the given point sets – in the following, we say that two sets are *x-separated* if they are separated by a plane orthogonal to the x-axis; we define *y-* and *z-separation* similarly. We first show how to handle the case when P and Q are both x- and y-separated by a simple direct argument:

▶ **Lemma 13.** *If P and Q are both x- and y-separated, then $\mathcal{S}_{\lll}(P, Q, R)$ has VC-dimension 1.*

Proof. Consider points $p, p' \in P$. Suppose $\{p\}$ and $\{p'\}$ are both shattered (by $\mathcal{S}_{\lll}(P, Q, R)$). Then there exist $r, r' \in R$ with $p \in N^2_{\lll}[r]$, $p' \notin N^2_{\lll}[r]$, $p' \in N^2_{\lll}[r']$, $p \notin N^2_{\lll}[r']$. Let $q, q' \in Q$ with $p \lhd_1 q \lhd_2 r$ and $p' \lhd_1 q' \lhd_2 r'$.

For any $\lhd_i$, we define $\lhd_{ix}$, $\lhd_{iy}$, and $\lhd_{iz}$ to be the x-, y-, and z-coordinate dominance relation, respectively. Because P and Q are x- and y-separated, we already know that $p \lhd_{1x} q'$, $p \lhd_{1y} q'$, and $p' \lhd_{1x} q$, $p' \lhd_{1y} q$. Note that $p \lhd_{1z} q'$ or $p' \lhd_{1z} q$, because otherwise, we must have $\lhd_{1z} \neq \top$ and $q' \lhd_{1z} p \lhd_{1z} q \lhd_{1z} p' \lhd_{1z} q'$: a contradiction. Thus, $p \lhd_1 q'$ or $p' \lhd_1 q$, implying $p \in N^2_{\lll}[r']$ or $p' \in N^2_{\lll}[r]$. ◀

We then handle the case when P and R are separated along all 3 axes.

▶ **Lemma 14.** *If P and R are x-, y-, and z-separated, then $\mathcal{S}_{\lll}(P, Q, R)$ has shatter dimension at most 6.*

Proof. Without loss of generality, say $P \subset (-\infty, 0)^3$ and $R \subset (0, \infty)^3$. Consider the following 6 set systems of the form $\mathcal{S}_{\lll}(P, Q \cap \Lambda, R)$, where Λ can be one of

$$((0,\infty) \times (0,\infty) \times \mathbb{R}), \quad (\mathbb{R} \times (0,\infty) \times (0,\infty)), \quad ((0,\infty) \times \mathbb{R} \times (0,\infty)),$$
$$((-\infty,0) \times (-\infty,0) \times \mathbb{R}), \quad (\mathbb{R} \times (-\infty,0)) \times (-\infty,0)), \quad ((-\infty,0) \times \mathbb{R} \times (-\infty,0)).$$

Each set system has VC-dimension at most 1 by Lemma 13 (or symmetric variants).

Each set in $\mathcal{S}_{\lll}(P, Q, R)$ can be expressed as the union of 6 sets (one of each) from these 6 set systems (since a point $q \in Q$ must has two positive coordinates or two negative coordinates). So, the number of distinct sets in $\mathcal{S}_{\lll}(P, Q, R)$ is at most $O(|P|^6)$. ◀

It remains to reduce the general case to the case when P and R are separated along all 3 axes. We accomplish this by a simple grid idea:

▶ **Lemma 15.** *For any $P, Q, R \subset \mathbb{R}^3$, $\mathcal{S}_{\lll}(P, Q, R)$ has shatter dimension at most* 51*.*

Proof. Build a (non-uniform) grid formed by the x-, y-, and z-coordinates of all the points of P. The grid has $O(|P|^3)$ cells. For each grid cell interior γ, consider the 8 octants at an arbitrary point inside γ; for each such octant τ, the set system $\mathcal{S}_{\lll}(P \cap \tau, Q, R \cap \gamma)$ has shatter dimension at most 6 by Lemma 14. For each $r \in R \cap \gamma$, the set $N^2_{\lll}[r]$ is the union of 8 sets (one of each) from the set systems for the 8 octants. So for each γ, the number of distinct sets in $\{N^2_{\lll}[r] : r \in R \cap \gamma\}$ is $O((|P|^6)^8)$, summing across all cells gives $|\{N^2_{\lll}[r] : r \in R\}| = O(|P|^3 \cdot (|P|^6)^8)$. ◀

We can now bound the shatter dimension for unit cubes.

▶ **Lemma 16.** *For any $P, Q, R \subset \mathbb{R}^3$ with $R \subset (0,1)^3$, write $N^2[r]$ as*

$$\Big\{p \in P : \exists q \in Q \text{ with } [\![q]\!] \text{ intersecting both } [\![p]\!] \text{ and } [\![r]\!]\Big\}.$$

Then the set system $(P, \{N^2[r] : r \in R\})$ has shatter dimension at most 37179*.*

Proof. Fix $\alpha = (\alpha_P, \alpha_Q) \in (\mathbb{Z}^3)^2$ with $\|\alpha_P - \alpha_Q\|_\infty \le 1$. For $p \in \alpha_P + (0,1)^3$ and $q \in \alpha_Q + (0,1)^3$, $[\![p]\!]$ intersects $[\![q]\!]$ iff $p - \alpha_P \lhd_1 q - \alpha_Q$, where we define $\lhd_{1x}$ to be $\succ_x$ if $x(\alpha_Q) = x(\alpha_P) + 1$, and $\prec_x$ if $x(\alpha_Q) = x(\alpha_P) - 1$, and $\top$ if $x(\alpha_Q) = x(\alpha_P)$; we define $\lhd_{1y}$ and $\lhd_{1z}$ in the same way. Similarly, for $q \in \alpha_Q + (0,1)^3$ and $r \in (0,1)^3$, $[\![q]\!]$ intersects $[\![r]\!]$ iff $q - \alpha_Q \lhd_2 r$ for some generalized dominance relation $\lhd_2$. Define the set system $\mathcal{S}_\alpha := \mathcal{S}_{\lll}\big(P \cap (\alpha_P + (0,1)^3) - \alpha_P, (Q \cap (\alpha_Q + (0,1)^3)) - \alpha_Q, R\big)$.

Each set $N^2[r]$ is the union of at most 27^2 sets (one of each) from the set systems $\mathcal{S}_\alpha$ for the at most 27^2 choices of α. By Lemma 15, we have $|\{N^2[r] : r \in R\}| = O((|P|^{51})^{27^2})$. ◀

▶ **Theorem 17.** *For any $P, Q, R \subset \mathbb{R}^3$, the set system $(P, \{N^2[r] : r \in R\})$ has shatter dimension at most* 37179*.*

Proof. Build a uniform grid of side length 1. For each grid cell $\alpha_R + (0,1)^3$ (with $\alpha_R \in \mathbb{Z}^3$), the number of different $N^2[r]$ sets over all $r \in R \cap (\alpha_R + (0,1)^3)$ is $O(|P \cap (\alpha_R + (-2,3)^3)|^{37179})$ by Lemma 16. Since each point $p \in P$ belongs to $O(1)$ number of expanded cells $\alpha_R + (-2,3)^3$, the sum over all α_R is $O(|P|^{37179})$. ◀

In the interest of simplicity, we have not optimized the above constant (which admittedly is quite large, but fortunately will not matter in our final algorithm).

3.2 Algorithm

Knowing that the VC-dimension is bounded, we could apply the framework of Chan et al. [8] to obtain a subquadratic algorithm for Diameter-2 of 3D unit cubes. However, the exponent would be very close to 2 (much worse than our more general result for arbitrary 3D boxes in Theorem 7, see the full version). We present a faster, direct algorithm that runs in *near-linear* time. The key observation is that although the final shatter dimension bound is large, the proof in Section 3.1 tells us that the set system is in some sense "made up of" a constant number of simpler subsystems with much smaller VC-dimension, namely, VC-dimension 1 (Lemma 13)! We can't just solve the problem for each subsystem separately – the diameter problem isn't decomposable that way. Instead, we "encode" each subsystem by adding a single coordinate value to each element, and in the end reduce the whole problem to an orthogonal range searching problem for vectors in a sufficiently large constant dimension. Due to lack of space, see the full version of the paper for the algorithm.

4 Subquadratic Diameter-3 Algorithm for 3D Unit Cubes

In this section, we present a subquadratic algorithm for testing whether the diameter of a 3D unit-cube graph is at most 3. This complements our lower bound results, which show conditionally that there are no similar Diameter-3 algorithms for 3D unit-ball graphs (Theorem 9) nor for 4D unit-hypercube graphs (Theorem 6). This result is more challenging than our result for diameter 2 in Section 3. As before, we warm up by studying the corresponding combinatorial problem of bounding the VC-dimension of the 3-neighborhoods of 3D unit-cube graphs (Section 4.1). The combinatorial proof will reveal a surprising connection to 2D pseudolines that we defer to the full paper.

Again, for a point $p \in \mathbb{R}^3$, recall that $[\![p]\!]$ denotes the unit cube centered at p. For simplicity, assume all coordinates are distinct. Like before, we will solve the problem in a slightly more general setting for 4 point sets P, Q, R, S, testing whether all distances between $[\![p]\!]$ and $[\![s]\!]$ for $(p, s) \in P \times S$ are at most 3 in the 4-partite intersection graphs of the unit cubes centered at P, Q, R, S.

4.1 VC-dimension bound

Following Section 3.1, we begin with the corresponding problem for generalized dominance relations. Let P, Q, R, S be four point sets in $\mathbb{R}^3$. Let $\lll = (\lhd_1, \lhd_2, \lhd_3)$ be three generalized dominance relations in $\mathbb{R}^3$. For each $s \in S$, we can write $N^3_{\lll}[s]$ as $\{p \in P : \exists (q, r) \in Q \times R$ with $p \lhd_1 q$ and $q \lhd_2 r$ and $r \lhd_3 s\}$. Define the set system $\mathcal{S}_{\lll}(P, Q, R, S) := (P, \{N^3_{\lll}[s] : s \in S\})$. We first prove that this set system has bounded shatter dimension.

As in Section 3.1, we do not use planarity arguments like previous proofs. Instead, we divide into cases based on separability of the given point sets. But as we go from 2- to 3-neighborhoods, we face more challenges. The first case turns out to be the most crucial (and interesting), where as the one that follows it is similar to Lemma 14.

▶ **Lemma 18.** *If P and Q are x-separated, Q and R are y-separated, and R and S are z-separated, then the set system $\mathcal{S}_{\lll}(P, Q, R, S)$ has VC-dimension at most 2.*

Proof. Suppose $p \lhd_j q$ iff $p \lhd_{jx} q$ and $p \lhd_{jy} q$ and $p \lhd_{jz} q$ with $\lhd_{jx} \in \{\prec_x, \succ_x, \top\}$, $\lhd_{jy} \in \{\prec_y, \succ_y, \top\}$, and $\lhd_{jz} \in \{\prec_z, \succ_z, \top\}$. We may assume that $\lhd_{1x} \neq \top$ (because if not, we can shift all the x-coordinates of P sufficiently far downward and replace $\lhd_{1x}$ with $\prec_x$), and similarly, all other $\lhd_{jx}, \lhd_{jy}, \lhd_{jz}$ are not $\top$. We first prove the following property:



Figure 1 In the yz plane, if p is dominated by q and p' is dominated by q', but p is not dominated by q' and p' is not dominated by q, then (i) and (ii) depict the only two possibilities.

Property: Let $p, p' \in P$ and $s, s' \in S$. If $p \in N^3_{\lll}[s]$, $p' \in N^3_{\lll}[s']$, $p \notin N^3_{\lll}[s']$, $p' \notin N^3_{\lll}[s]$, then $p \prec_y p'$ implies $s' \blacktriangleleft_y s$, where $\blacktriangleleft_y \in \{\prec_y, \succ_y\}$ is a relation determined solely from the choices of $\lhd_1, \lhd_2, \lhd_3$.

Proof: Let $q, q' \in Q$ and $r, r' \in R$ with $p \lhd_1 q \lhd_2 r \lhd_3 s$ and $p' \lhd_1 q' \lhd_2 r' \lhd_3 s'$. We may assume neither $p \lhd_1 q'$ nor $p' \lhd_1 q$, because otherwise, $p \in N^3_{\lll}[s']$ or $p' \in N^3_{\lll}[s]$. Because P and Q are x-separated, we already know $p \lhd_{1x} q'$ and $p' \lhd_{1x} q$. So, there are two remaining possibilities: (i) $p \lhd_{1y} q \lhd_{1y} p' \lhd_{1y} q'$ and $p' \lhd_{1z} q' \lhd_{1z} p \lhd_{1z} q$, or (ii) $p' \lhd_{1y} q' \lhd_{1y} p \lhd_{1y} q$ and $p \lhd_{1z} q \lhd_{1z} p' \lhd_{1z} q'$. (See Figure 1.)

Define $\blacktriangleleft_{1z} \in \{\prec_z, \succ_z\}$ to be $\lhd_{1z}$ iff $\lhd_{1y}$ is $\prec_y$. Then $p \prec_y p'$ implies $q' \blacktriangleleft_{1z} q$.

Similarly, we may assume neither $q \lhd_2 r'$ nor $q' \lhd_2 r$, because otherwise, $p \in N^3_{\lll}[s']$ or $p' \in N^3_{\lll}[s]$. Because Q and R are y-separated, $q \lhd_{2y} r'$ and $q' \lhd_{2y} r$. There are two remaining possibilities: (i) $q \lhd_{2z} r \lhd_{2z} q' \lhd_{2z} r'$ and $q' \lhd_{2x} r' \lhd_{2x} q \lhd_{2x} r$, or (ii) $q' \lhd_{2z} r' \lhd_{2z} q \lhd_{2z} r$ and $q \lhd_{2x} r \lhd_{2x} q' \lhd_{2x} r'$.

Define $\blacktriangleleft_{2x} \in \{\prec_x, \succ_x\}$ to be $\lhd_{2x}$ iff $\lhd_{2z}$ is $\blacktriangleleft_{1z}$. Then $q' \blacktriangleleft_{1z} q$ implies $r \blacktriangleleft_{2x} r'$.

Similarly (and lastly), we may assume neither $r \lhd_3 s'$ nor $r' \lhd_3 s$, because otherwise, $p \in N^3_{\lll}[s']$ or $p' \in N^3_{\lll}[s]$. Because R and S are z-separated, $r \lhd_{3z} s'$ and $r' \lhd_{3z} s$. There are two remaining possibilities: (i) $r \lhd_{3x} s \lhd_{3x} r' \lhd_{3x} s'$ and $r' \lhd_{3y} s' \lhd_{3y} r \lhd_{3y} s$, or (ii) $r' \lhd_{3x} s' \lhd_{3x} r \lhd_{3x} s$ and $r \lhd_{3y} s \lhd_{3y} r' \lhd_{3y} s'$.

Define $\blacktriangleleft_y \in \{\prec_y, \succ_y\}$ to be $\lhd_{3y}$ iff $\lhd_{3x}$ is $\blacktriangleleft_{2x}$. Then $r \blacktriangleleft_{2x} r'$ implies $s' \blacktriangleleft_y s$. □

Now, consider 3 points $p, p', p'' \in P$, with $p \prec_y p' \prec_y p''$. Suppose $\{p, p''\}$ and $\{p'\}$ are both shattered. Then there exist $s, s' \in S$ with $p, p'' \in N^3_{\lll}[s]$, $p' \notin N^3_{\lll}[s]$, $p' \in N^3_{\lll}[s']$, $p, p'' \notin N^3_{\lll}[s']$. Applying the property twice yields $s' \blacktriangleleft_y s$ and $s \blacktriangleleft_y s'$: a contradiction. ◀

▶ **Lemma 19.** *If (a) P, Q are both x- and y-separated, or (b) Q, R are both x- and y-separated, or (c) R, S are both x- and y-separated, then $\mathcal{S}_{\lll}(P, Q, R, S)$ has VC-dimension 1.*

Proof. Consider 2 points $p, p' \in P$. Suppose $\{p\}$ and $\{p'\}$ are both shattered. Then there exist $s, s' \in S$ with $p \in N^3_{\lll}[s]$, $p' \notin N^3_{\lll}[s]$, $p' \in N^3_{\lll}[s']$, $p \notin N^3_{\lll}[s']$. Let $q, q' \in Q$ and $r, r' \in R$ with $p \lhd_1 q \lhd_2 r \lhd_3 s$ and $p' \lhd_1 q' \lhd_2 r' \lhd_3 s'$.

For (a), $p \lhd_{1x} q'$, $p \lhd_{1y} q'$, and $p' \lhd_{1x} q$, $p' \lhd_{1y} q'$ by the separation assumptions. Note that $p \lhd_{1z} q'$ or $p' \lhd_{1z} q$, because otherwise, $q' \lhd_{1z} p \lhd_{1z} q \lhd_{1z} p' \lhd_{1z} q'$: a contradiction. Thus, $p \lhd_1 q'$ or $p' \lhd_1 q$, implying $p \in N^3_{\lll}[s']$ or $p' \in N^3_{\lll}[s]$.

Claims (b) and (c) follow from a similar argument. ◀

We now handle the case when P and S are separated along all 3 axes, by invoking the previous cases some constant number of times:

▶ **Lemma 20.** *If P and S are x-, y-, and z-separated, then $\mathcal{S}_{\lll}(P, Q, R, S)$ has shatter dimension at most* 21*.*

Proof. Denote $\mathbb{R}_+$ and $\mathbb{R}_-$ to be $(0, \infty)$ and $(-\infty, 0)$, respectively. Without loss of generality, say $P \subset \mathbb{R}^3_-$ and $S \subset \mathbb{R}^3_+$. Consider the following 15 set systems.

$$\begin{aligned}
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R}_+ \times \mathbb{R}_- \times \mathbb{R}), \quad R \cap (\mathbb{R} \times \mathbb{R}_+ \times \mathbb{R}_-), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R} \times \mathbb{R}_+ \times \mathbb{R}_-), \quad R \cap (\mathbb{R}_- \times \mathbb{R} \times \mathbb{R}_+), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R}_- \times \mathbb{R} \times \mathbb{R}_+), \quad R \cap (\mathbb{R}_+ \times \mathbb{R}_- \times \mathbb{R}), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R}_- \times \mathbb{R}_+ \times \mathbb{R}), \quad R \cap (\mathbb{R}_+ \times \mathbb{R} \times \mathbb{R}_-), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R} \times \mathbb{R}_- \times \mathbb{R}_+), \quad R \cap (\mathbb{R}_- \times \mathbb{R}_+ \times \mathbb{R}), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R}_+ \times \mathbb{R} \times \mathbb{R}_-), \quad R \cap (\mathbb{R} \times \mathbb{R}_- \times \mathbb{R}_+), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}), \quad R, \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R}_+ \times \mathbb{R} \times \mathbb{R}_+), \quad R, \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R} \times \mathbb{R}_+ \times \mathbb{R}_+), \quad R, \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R}_- \times \mathbb{R}_- \times \mathbb{R}), \quad R \cap (\mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R}_- \times \mathbb{R} \times \mathbb{R}_-), \quad R \cap (\mathbb{R}_+ \times \mathbb{R} \times \mathbb{R}_+), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q \cap (\mathbb{R} \times \mathbb{R}_- \times \mathbb{R}_-), \quad R \cap (\mathbb{R} \times \mathbb{R}_+ \times \mathbb{R}_+), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q, \quad R \cap (\mathbb{R}_- \times \mathbb{R}_- \times \mathbb{R}), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q, \quad R \cap (\mathbb{R}_- \times \mathbb{R} \times \mathbb{R}_-), \quad S)\\
&\mathcal{S}_{\lll}(P, \quad Q, \quad R \cap (\mathbb{R} \times \mathbb{R}_- \times \mathbb{R}_-), \quad S).
\end{aligned}$$

The first 6 set systems have VC-dimension at most 2 by Lemma 18 (or symmetric variants); the next 9 have VC-dimension 1 by Lemma 19 (or symmetric variants).

Each set in $\mathcal{S}_{\lll}(P, Q, R, S)$ can be expressed as the union of 15 sets one from each of these 15 set systems: this is because, in any point sequence $\langle p, q, r, s\rangle$ with $p \in (-\infty, 0)^3$ and $s \in (0, \infty)^3$, there must be a consecutive pair going from negative to positive x-coordinate, and a consecutive pair going from negative to positive y-coordinate, and a consecutive pair going from negative to positive z-coordinate; all 3 such pairs could be distinct, or 2 of them could be the same. So, $|\mathcal{S}_{\lll}(P, Q, R, S)| = O((|P|^2)^6 \cdot |P|^9)$. ◀

We can now reduce the general case to the case when P and S are separated along all 3 axes.

▶ **Lemma 21.** *For any $P, Q, R, S \subset \mathbb{R}^3$, $\mathcal{S}_{\lll}(P, Q, R, S)$ has shatter dimension ≤ 171.*

Proof. Build a (non-uniform) grid formed by the x-, y-, and z-coordinates of all the points of P. The grid has $O(|P|^3)$ cells. For each grid cell γ, consider the 8 octants at an arbitrary point inside γ; for each such octant τ, the set system $\mathcal{S}_{\lll}(P \cap \tau, Q, R, S \cap \gamma)$ has shatter dimension at most 18 by Lemma 20. For each $s \in S \cap \gamma$, the set $N^3_{\lll}[s]$ is the union of 8 sets (one of each) from the set systems for the 8 octants. So for each γ, the number of distinct sets in $\{N^3_{\lll}[s] : s \in S \cap \gamma\}$ is $O((|P|^{21})^8)$. Thus, we have $|\{N^3_{\lll}[s] : s \in S\}| = O(|P|^3 \cdot (|P|^{21})^8)$. ◀

As before, unit cubes reduce to generalized dominance relations:

▶ **Lemma 22.** *For any $P, Q, R, S \subset \mathbb{R}^3$ with $S \subset (0, 1)^3$, write $N^3[s]$ as*

$$\Big\{p \in P : \exists (q, r) \in Q \times R \text{ s.t. } [\![p]\!] \text{ intersects } [\![q]\!], [\![q]\!] \text{ intersects } [\![r]\!], \text{ and } [\![r]\!] \text{ intersects } [\![s]\!]\Big\}.$$

Then the set system $(P, \{N^3[s] : s \in S\})$ has shatter dimension at most 3365793*.*

Proof. Fix $\alpha = (\alpha_P, \alpha_Q, \alpha_R) \in (\mathbb{Z}^3)^3$ with $\|\alpha_R\|_\infty, \|\alpha_Q - \alpha_R\|_\infty, \|\alpha_P - \alpha_Q\|_\infty \le 1$. For $p \in \alpha_P + (0,1)^3$ and $q \in \alpha_Q + (0,1)^3$, $[\![p]\!]$ intersects $[\![q]\!]$ iff $p - \alpha_P \lhd_1 q - \alpha_Q$, for some generalized dominance relation $\lhd_1$ as in the proof of Lemma 16. Similarly, for $q \in \alpha_Q + (0,1)^3$ and $r \in \alpha_R + (0,1)^3$, $[\![q]\!]$ intersects $[\![r]\!]$ iff $q - \alpha_Q \lhd_2 r - \alpha_R$ for some generalized dominance relation $\lhd_2$. Similarly, for $r \in \alpha_R + (0,1)^3$ and $s \in (0,1)^3$, $[\![r]\!]$ intersects $[\![s]\!]$ iff $r - \alpha_R \lhd_3 s$ for some generalized dominance relation $\lhd_3$. Define the set system $\mathcal{S}_\alpha = \mathcal{S}_{\lll}(P \cap (\alpha_P + (0,1)^3) - \alpha_P, (Q \cap (\alpha_Q + (0,1)^3)) - \alpha_Q, (R \cap (\alpha_R + (0,1)^3)) - \alpha_R, S)$.

Each set $N^3[s]$ is the union of at most 27^3 sets (one of each) from the set systems $\mathcal{S}_\alpha$ for the at most 27^3 choices of α. By Lemma 21, we have $|\{N^3[s] : s \in S\}| = O((|P|^{171})^{27^3})$. ◀

▶ **Theorem 23.** *For any $P, Q, R, S \subset \mathbb{R}^3$, the set system $(P, \{N^3[s] : s \in S\})$ has shatter dimension at most* 3365793*.*

Proof. Build a uniform grid of side length 1. For each grid cell $\alpha_S + (0,1)^3$ (with $\alpha_S \in \mathbb{Z}^3$), the number of different $N^3[s]$ sets over all $s \in S \cap (\alpha_S + (0,1)^3)$ is $O(|P \cap (\alpha_S + (-3,4)^3)|^{3365793})$ by Lemma 22. Since each point $p \in P$ belongs to $O(1)$ number of expanded cells $\alpha_S + (-3,4)^3$, the sum over all α_S is $O(|P|^{3365793})$. ◀

4.2 Algorithm

Knowing that the VC-dimension is bounded, we could at this point apply the framework of [8] to obtain a subquadratic algorithm for Diameter-3 in 3D unit cubes. However, the exponent would be *extremely* close to 2 (around $2 - 1/(4 \cdot 3365794) \approx 1.99999993$). We present a faster, direct algorithm with a more reasonable exponent smaller than 2. The key observation is that although the final shatter dimension bound is large, the proof in Section 4.1 tells us that the set system is in some sense "made up of" a (very large but) constant number of simpler subsystems with much smaller VC-dimension, of 1 and 2 (Lemma 18 and Lemma 19). As in our Diameter-2 algorithm, we can't just solve the problem for each subsystem separately, because the diameter problem isn't decomposable. Instead, we "encode" each subsystem as an extra "2-dimensional constraint", and in the end reduce the whole problem to a "multi-level" range searching problem in a sufficiently large constant dimension.

Multi-level range searching [2, 18] is usually solved by sampling-based geometric divide-and-conquer techniques, e.g., via so-called "cuttings", but analogs of cuttings provably do not exist for abstract set systems, even with VC-dimension 2 (e.g., see [3, Remark 4.7]). Fortunately, our proof of VC-dimension 2 in the full paper reveals a stronger property, namely, that the 3-neighborhoods there actually form a *point-pseudoline system*. (It is quite unexpected that 3D unit cubes could naturally produce pseudoline arrangements.) Cuttings and multi-level range searching exists for pseudolines. Due to lack of space, details of the algorithm are given in the full paper.

References

1 A. Karim Abu-Affash, Paz Carmi, Anil Maheshwari, Pat Morin, Michiel Smid, and Shakhar Smorodinsky. Approximating maximum diameter-bounded subgraph in unit disk graphs. *Discret. Comput. Geom.*, 66(4):1401–1414, 2021. doi:10.1007/s00454-021-00327-y.

2 Pankaj K Agarwal. Range searching. In *Handbook of discrete and computational geometry*, pages 1057–1092. Chapman and Hall/CRC, 2017. doi:10.1201/9781315119601.

3 Sujoy Bhore, Timothy M. Chan, Zhengcheng Huang, Shakhar Smorodinsky, and Csaba D. Tóth. Sparse bounded hop-spanners for geometric intersection graphs. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry, SoCG 2025, Kanazawa, Japan, June 23-27, 2025*, volume 332 of *LIPIcs*, pages 17:1–17:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SOCG.2025.17.

4 Nicolas Bousquet and Stéphan Thomassé. VC-dimension and Erdős–Pósa property. *Discret. Math.*, 338(12):2302–2317, 2015. doi:10.1016/j.disc.2015.05.026.

5 Karl Bringmann, Sándor Kisfaludi-Bak, Marvin Künnemann, André Nusser, and Zahra Parsaeian. Towards sub-quadratic diameter computation in geometric intersection graphs. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry, SoCG 2022, Berlin, Germany, June 7-10, 2022*, volume 224, pages 21:1–21:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.21.

6 Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Trans. Algorithms*, 15(2):21:1–21:38, December 2019. doi:10.1145/3218821.

7 Timothy M. Chan. Finding triangles and other small subgraphs in geometric intersection graphs. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1777–1805. SIAM, 2023. doi:10.1137/1.9781611977554.CH68.

8 Timothy M. Chan, Hsien-Chih Chang, Jie Gao, Sándor Kisfaludi-Bak, Hung Le, and Da Wei Zheng. Truly subquadratic time algorithms for diameter and related problems in graphs of bounded VC-dimension. In *66th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 2728–2765, 2025. doi:10.48550/arXiv.2510.16346.

9 Timothy M. Chan, Hsien-Chih Chang, Jie Gao, Sándor Kisfaludi-Bak, Hung Le, and Da Wei Zheng. Charting the diameter computation landscape on intersection graphs in 2D, 2026. Under preparation.

10 Timothy M. Chan, Qizheng He, and Yuancheng Yu. On the fine-grained complexity of small-size geometric set cover and discrete k-center for small k. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, Paderborn, Germany, July 10-14, 2023*, volume 261 of *LIPIcs*, pages 34:1–34:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.34.

11 Hsien-Chih Chang, Jie Gao, and Hung Le. Computing diameter+2 in truly-subquadratic time for unit-disk graphs. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry, SoCG 2024, Athens, Greece, June 11-14, 2024*, volume 293, pages 38:1–38:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. Arxiv version https://arxiv.org/abs/2401.12881. doi:10.4230/LIPIcs.SoCG.2024.38.

12 Victor Chepoi, Bertrand Estellon, and Yann Vaxès. Covering planar graphs with a fixed number of balls. *Discret. Comput. Geom.*, 37(2):237–244, 2007. doi:10.1007/s00454-006-1260-0.

13 Guillaume Ducoffe, Michel Habib, and Laurent Viennot. Diameter, eccentricities and distance oracle computations on H-minor free graphs and graphs of bounded (distance) vapnik-chervonenkis dimension. *SIAM J. Comput.*, 51(5):1506–1534, 2022. doi:10.1137/20M136551X.

14 Lech Duraj, Filip Konieczny, and Krzysztof Potępa. Better diameter algorithms for bounded VC-dimension graphs and geometric intersection graphs. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, Royal Holloway, London, United Kingdom, September 2-4, 2024*, volume 308, pages 51:1–51:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.51.

15 Paweł Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{O}(n^{5/3})$ time. *SIAM J. Comput.*, 50(2):509–554, 2021. doi:10.1137/18M1193402.

16 Hung Le and Christian Wulff-Nilsen. VC set systems in minor-free (di)graphs and applications. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 5332–5360. SIAM, 2024. doi:10.1137/1.9781611977912.192.

17 Jason Li and Merav Parter. Planar diameter via metric compression. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 152–163, Phoenix, AZ, USA and New York, NY, USA, 2019. ACM. doi:10.1145/3313276.3316358.

18 Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discret. Comput. Geom.*, 10:157–182, 1993. doi:10.1007/BF02573972.

19 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 515–524. ACM, 2013. doi:10.1145/2488608.2488673.

20 Norbert Sauer. On the density of families of sets. *J. Comb. Theory A*, 13(1):145–147, 1972. doi:10.1016/0097-3165(72)90019-2.

21 Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1):247–261, 1972. doi:10.2140/pjm.1972.41.247.

Computing the Girth of a Segment Intersection Graph

Timothy M. Chan
Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign, IL, USA

Yuancheng Yu
Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign, IL, USA

Abstract

We present an algorithm that computes the girth of the intersection graph of n given line segments in the plane in $O(n^{1.483})$ expected time. This is the first such algorithm with $O(n^{3/2-\varepsilon})$ running time for a positive constant ε, and makes progress towards an open question posed by Chan (SODA 2023). The main techniques include (i) the usage of recent subcubic algorithms for bounded-difference min-plus matrix multiplication, and (ii) an interesting variant of the planar graph separator theorem. The result extends to intersection graphs of connected algebraic curves or semialgebraic sets of constant description complexity.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geometric intersection graphs, girth, shortest paths, graph separators, matrix multiplication

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.30

Related Version *Full Version*: `https://arxiv.org/pdf/2603.21585`

Funding *Timothy M. Chan*: Work supported by NSF Grant CCF-2224271.

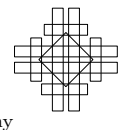
1 Introduction

Intersection graphs of geometric objects have received considerable attention in computational geometry. They generalize planar graphs (every planar graph is an intersection graph of disks in 2D by the Koebe–Andreev–Thurston theorem), and they arise in numerous applications (for example, unit disk graphs can be used to model communication networks, and intersection graphs of line segments in 2D can be used to model road networks). Over the years, researchers have studied many standard computational problems on geometric intersection graphs, aiming for algorithms that run faster than those for general graphs. The problems considered include the computation of connected components [1], all-pairs shortest paths [17], diameter [10, 15], minimum cut [12], maximum matching [6], and finding small-sized subgraphs (such as triangles, cycles of a fixed length, cliques of a fixed size, etc.) [14].

In this paper, we focus on another of the most basic graph algorithm problems, which has not been as well explored in the context of geometric intersection graphs: the computation of the *girth*, i.e., the length of the shortest cycle (see Figure 1). Girth is related to various topics in graph theory, including the chromatic number, treewidth, diameter, Ramsey theory, etc. [5, 22, 39, 27, 18, 23]. It can be computed in $O(mn)$ or $O(n^\omega)$ time for general unweighted undirected graphs with n vertices and m edges [32], where $\omega < 2.371339$ is the matrix multiplication exponent [2]. For the case of planar graphs, a series of papers [24, 25, 13, 46] culminated in a linear-time algorithm by Chang and Lu [19]. However, for intersection graphs, we are aware of only two main algorithmic results:

1. Kaplan et al. at ESA'19 [33] gave an $O(n \log n)$-time algorithm for computing the girth of the intersection graph of n disks in 2D (in the case of unit disks, this was subsequently improved by Klost and Mulzer [35] to an $O(n)$-time algorithm which does not require the disks be given, just the graph).


© Timothy M. Chan and Yuancheng Yu;
licensed under Creative Commons License CC-BY 4.0



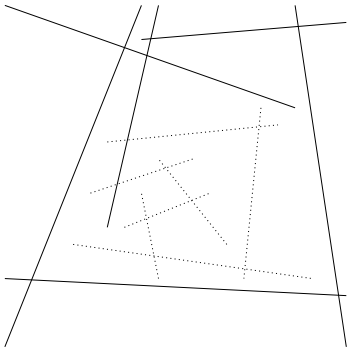



Figure 1 An example of the shortest cycle of a segment intersection graph (segments not in the shortest cycle are dotted). The girth is 6.

2. Chan at SODA'23 [14] gave an $\widetilde{O}(n^{3/2})$-time[1] algorithm for computing the girth of the intersection graph of n line segments in 2D.

Chan's work naturally raises the following question:

Is there an $O(n^{3/2-\varepsilon})$-time algorithm for computing the girth of the intersection graph of n line segments for some constant $\varepsilon > 0$?

The case of line segments is particularly important, not only because of direct relevance to road networks, for example, but also because it serves as a stepping stone towards the general case of algebraic curves or semialgebraic sets in the plane.

1.1 New result

We answer the above open question in the affirmative by presenting the first algorithm that breaks the $n^{3/2}$ barrier. Specifically, our algorithm computes the girth for the intersection graph of n line segments in 2D in $O(n^{1.483})$ expected time. The algorithm also extends to intersection graphs of algebraic curves or semialgebraic sets where each object is connected and has constant description complexity – essentially, most families of constant-size objects in 2D that are commonly encountered.

1.2 Why is the problem challenging?

First note that Kaplan et al.'s approach [33] is not applicable for our problem, as it exploited special properties of disks that do not hold for line segments (namely, that any intersection graph of disks with girth more than 3 must be planar). For the related problem of finding short cycles, Chan [14] extended Kaplan et al.'s result to arbitrary fat objects in any constant dimension, by using a shifted quadtree approach, but line segments are not fat.

For intersection graphs of line segments, Chan [14] has already given an $O(n^{1.408})$-time algorithm for detecting a triangle (i.e., 3-cycle), and an $\widetilde{O}(n)$-time algorithm for detecting a 4-cycle. Thus, it suffices to handle the remaining case when the girth is greater than 4. It is known that any intersection graph of n line segments without 4-cycles must be sparse, i.e., has $O(n)$ edges. We can explicitly construct the arrangement of the line segments and form a planar graph of $O(n)$ size whose vertices are the endpoints and intersection points. This

[1] The $\widetilde{O}$ notation hides polylogarithmic factors.

way, our problem is reduced to a variant of the girth problem on a planar graph, where the cost of a cycle is the number of turns – turning at an intersection point has unit cost, but going straight through an intersection has zero cost.

However, this variant does not seem directly reducible to the standard girth problem on planar graphs. One could create a gadget at each intersection point and obtain a graph with 0/1 edge weights, such that the weight of a path in the new graph corresponds to the number of turns along the path. But the resulting graph would no longer be planar (it would be 1-*planar* instead); besides, unintended small cycles would be introduced in such a gadget. In addition, many properties about shortest paths in planar graphs no longer hold. For example, two shortest paths in a planar graph cannot cross more than once (if ties are broken carefully), but two minimum-turn paths may cross numerous times. Consequently, many techniques devised for shortest-path-related problems in planar graphs (e.g., based on some kind of Monge property [29], or abstract Voronoi diagrams [11, 30], or VC-dimension arguments [37, 26, 36, 15]) seem to fall apart for our problem.

One technique that still works is *separators*. In fact, Chan's previous $\widetilde{O}(n^{3/2})$-time algorithm is obtained by a simple application of separators. We can divide the graph into two parts by a separator of size $O(\sqrt{n})$, and recurse on both parts. This handles the case when the shortest cycle lies completely in one part. To deal with the remaining case when the shortest cycle passes through a separator vertex, we can run BFS from each of the $O(\sqrt{n})$ separator vertices, in $\widetilde{O}(n^{3/2})$ total time.

For the case when the girth is a constant (or for detecting a cycle of a fixed constant length), Chan [14] has shown how to adapt the separator approach to obtain a faster algorithm running in $\widetilde{O}(n^{\omega/2}) \leq O(n^{1.186})$ time. The main observation is that the $O(\sqrt{n})$ calls to BFS is overkill – we only need to compute shortest path distances with *both* sources and destinations from the $O(\sqrt{n})$ separator vertices. We can compute such distances recursively as we compute the girth. Combining the output of the two recursive calls involves computing min-plus products of $O(\sqrt{n}) \times O(\sqrt{n})$ distance matrices. When the matrix entries are integers bounded by a constant, min-plus product reduces to $O(1)$ number of standard (Boolean) matrix product. This is roughly how the $\widetilde{O}(n^{\omega/2})$ time bound was obtained.

For larger girth, however, this approach does not work well, because the complexity of min-plus product grows for larger integer values. (Another issue is the use of *color-coding* [3] in Chan's cycle detection algorithm [14], which led to exponential dependence on the cycle length.)

On the other hand, when the girth g is very large, one could obtain an $\widetilde{O}(n^2/g)$-time algorithm by a well-known hitting set technique [44, 47] (there exists a subset of $\widetilde{O}(n/g)$ vertices which hit all shortest paths of length $\Omega(g)$). The main difficulty lies in the intermediate case, when the girth is not too small and not too large.

1.3 Our approach

Ingredient 1: Bounded-difference min-plus matrix multiplication. The previous separator-based approach faces an $n^{3/2}$ barrier because min-plus matrix multiplication is widely conjectured to require near-cubic time (according to the "All-Pairs Shortest Paths (APSP) hypothesis" from fine-grained complexity [45]). Our first idea is to employ recent results which solve certain powerful special cases of min-plus product in truly subcubic time – namely, the case when the input matrices are integer-valued and satisfy a *bounded-difference* property, i.e., the difference between any two adjacent entries (in each row or in each column) is at most a constant. Chan and Lewenstein [16] gave the first truly subquadratic algorithm for the related min-plus convolution problem for bounded-difference sequences,

while Bringmann et al. [9] gave the first truly subcubic algorithm for bounded-difference min-plus product. Bringmann et al.'s original algorithm ran in $O(n^{2.8244})$ time, and after a series of improvements [31, 20], Chi et al. [21] obtained the current best running time of $\widetilde{O}(n^{(3+\omega)/2}) \leq O(n^{2.686})$ (see Section 2.1).

Bounded-difference min-plus convolution and matrix multiplication have recently been applied to solve a variety of problems from different domains, including language edit distance and RNA folding [9], single-source replacement paths and Range Mode [31], tree edit distance [40], approximate APSP [43], knapsack [7, 8], etc. Our algorithm, being one of the first applications to a computational geometry problem, is thus noteworthy.

Ingredient 2: A variant of the planar-graph separator theorem. Why is the bounded-difference property relevant in our application? Suppose the separator S is connected, e.g., is a cycle (as in Miller's version of the separator theorem [41]). Then for any vertex q and two adjacent separator vertices $s, s' \in S$, $d(s, q)$ and $d(s', q)$ differ by at most 1 due to the triangle inequality. Thus, the distance matrices we encounter during recursion would indeed satisfy the bounded-difference property.

Unfortunately, cycle separators do not always exist when the given planar graph is not 2-connected. Our key idea is to use an interesting variant of the planar-graph separator theorem, which holds for any planar graphs: given any $\alpha > 0$, there always exists a separator S consisting of two subsets P and Q, where

- the first subset P has size $O(n^{1/2+\alpha})$ but has at most 2 connected components (in fact, at most 2 paths);
- the second subset Q may not be connected but has size $O(n^{1/2-\alpha})$.

We are not aware of this variant of the separator theorem explicitly stated before (though we will not be surprised if it appeared before), but it follows by straightforward modification of standard proofs. See Section 2.2 for the precise statement and proof.

This variant of separators is exactly the tool we need to break the $n^{3/2}$ barrier: since P has only $O(1)$ connected pieces, we can use bounded-difference min-plus products to compute distances involving P faster; on the other hand, since Q has size $\ll \sqrt{n}$, we can afford to compute distances involving Q more naively. We choose the trade-off parameter α carefully to balance cost. This is the main high-level idea of our algorithm. Quite a bit of technical effort, however, is needed to make the idea work (for example, the number of connected components may grow as we recurse, but not by much, as we will show, and we eventually need two separate recursive algorithms, one for computing distances and one for computing the girth). See Sections 3–4 for the detailed algorithm and analysis.

Ingredient 3: Ensuring simplicity of walks. When computing a shortest path between two vertices, we do not need to worry about ensuring simplicity of the path, since a shortest walk must automatically be a shortest path (if a vertex is repeated, we can take a short-cut to obtain a shorter walk). However, more care is needed for the shortest cycle problem in undirected graphs. When concatenating several shortest paths to form a candidate cycle (via min-plus products of several distance matrices), one potential issue is that the resulting closed walk might become trivial (we might be completely retracing our steps, so that short-cutting yields an empty walk!). The earlier algorithm by Chan [14] for detecting constant-length cycles used the color-coding technique [3] to ensure simplicity, but unfortunately color-coding yields exponential dependence on the cycle length. For general graphs, there are other strategies to ensure simplicity, (e.g., by computing shortest paths in different subgraphs with some vertices and edges removed, or with directions added, so as to avoid concatenating two

identical shortest paths). But ensuring simplicity is trickier in our setting, since we need to retain the bounded-difference property if we work with new subgraphs. Still, we have found one way to deal with this technical issue, using ideas given in Section 2.3.

2 Preliminaries

Before presenting our algorithm, we begin by providing more details on the three ingredients mentioned in the introduction.

Let $\delta_G(u, v)$ denote the shortest-path length between u and v in an unweighted undirected graph G. Let $\delta_G[A, B]$ denote the distance matrix $\{\delta_G(u, v)\}_{u\in A, v\in B}$ for two vertex subsets A and B in a graph G. Let $G[S]$ denote the subgraph of G induced by a vertex subset S.

2.1 Bounded-difference min-plus product

Given two matrices A, B, their *min-plus product* $A \star B$ is defined as

$$(A \star B)(i, j) = \min_k (A(i, k) + B(k, j)).$$

As mentioned, we will use a known subcubic algorithm for computing the min-plus product of two $n \times n$ integer matrices that satisfy a bounded-difference property. The following is due to Chi et al. [21]:

▶ **Lemma 1.** *Given two integer $n \times n$ matrices A and B where each column of A satisfies the bounded-difference property (i.e., $|A(i, j) - A(i + 1, j)| \leq O(1)$ for every i, j), we can compute their min-plus product $A \star B$ in $\widetilde{O}(n^{(3+\omega)/2}) \leq O(n^{2.686})$ expected time.*

The same holds if instead each row of A satisfies the bounded-difference property (i.e., $|A(i, j) - A(i, j + 1)| \leq O(1)$), or if the property holds for B rather than A.

2.2 A variant of the planar separator theorem

The second ingredient is an interesting variant of the planar-graph separator theorem, in which the separator is divided into two subsets P and Q: P may be larger but has only 2 connected pieces, while Q may not be connected but is smaller. The precise statement is given below (the original theorem corresponds to the case when $p = \sqrt{n}$ with unit loads, while our application requires the case when $p \gg \sqrt{n}$). The proof follows by modifying standard proofs [38, 34, 28].

▶ **Lemma 2.** *Let $G = (V, E)$ be a planar graph with n vertices, where each vertex has a nonnegative* weight *and a nonnegative* load*, with total weight W, total load Λ, and the maximum individual vertex weight is at most εW for a sufficiently small constant $\varepsilon > 0$. For a given parameter p, we can partition V into subsets V_1, V_2, S, in $O(n)$ time, such that*

- *the total weight of V_i is at most $3W/4$ for each $i \in \{1, 2\}$;*
- *there are no edges in $V_1 \times V_2$;*
- *S can be further partitioned into P and Q such that P is the union of at most two paths of length $O(p)$, and Q has total load at most $O(\Lambda/p)$.*

Proof. We first assume that G is connected. Let $\mathcal{T}$ be a BFS tree of G from a fixed source vertex. For any $uv \notin \mathcal{T}$, let $\text{cycle}(\mathcal{T}, uv)$ denote the unique cycle in $\mathcal{T} \cup \{uv\}$ (called the *fundamental cycle*), which consists of uv and the unique path in $\mathcal{T}$ connecting u and v.

Let Σ be a (combinatorial) triangulation of $\mathcal{T}$. Define face weights of Σ by examining each vertex v, picking an arbitrary incident face f, and adding v's weight to f. Then the total face weight is W. Consider the dual tree with respect to $\mathcal{T}$ in Σ (which has maximum degree 3).

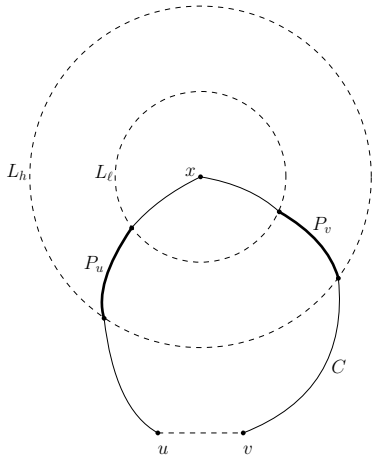


Figure 2 Proof of Lemma 2. Solid curves consist of edges in G, whereas dashed curves may contain edges in the triangulation but not in G.

By standard results on tree separators [34, 28]. we can find an edge which separates this dual tree into two parts each with total face weight at most $3W/4$. In Σ, this corresponds to an edge uv such that either side of $C = \text{cycle}(\mathcal{T}, uv)$ has total face weight at most $3W/4$; this also bounds the total weight of the vertices strictly inside/outside C. If $|C| \leq p$ then the desired separator can obtained by setting $P = C$ and $Q = \emptyset$. Below we may assume $|C| > p$.

Let x be the lowest common ancestor of u, v in $\mathcal{T}$, and $\mathcal{T}'$ be the subtree of $\mathcal{T}$. Let $L_i \subset V$ be the set of vertices in level i of $\mathcal{T}'$ (with $L_0 = \emptyset$). Let m be the smallest integer such that the total weight of vertices in the first m levels of $\mathcal{T}'$ is at least $W/2$. By the pigeonhole principle, there exists $\ell \in [m - p, m)$ such that L_ℓ has total load at most Λ/p. Similarly there exists $h \in (m, m + p]$ such that L_h has total load at most Λ/p.

If $V_{\leq \ell} = \bigcup_{i=1}^{\ell} L_i$ has weight at least $W/4$, then since $V_{\leq \ell}$ has weight at most $W/2$ by definition of m, the desired separator can be obtained by setting $Q = L_\ell$ and $P = \emptyset$.

Similarly, if $V_{\geq h} = \bigcup_{i=h}^{\infty} L_i$ has weight at least $W/4$, then the desired separator can be obtained by setting $Q = L_h$ and $P = \emptyset$.

In the remaining case, $V_m = \bigcup_{i=\ell+1}^{h-1} L_i$ has weight at least $W/2$. Let $V_{\text{in}}, V_{\text{out}}$ be the set of vertices in V_m that are strictly inside or outside C, respectively. Without loss of generality, say V_{in} has weight at least $W/4$. Recall that V_{in} has weight at most $3W/4$ by definition of C. Let P_u (resp. P_v) be the portion of the shortest path from x to u (resp. v) with levels between ℓ and h. Then P_u and P_v are paths of length $O(p)$. The desired separator can be obtained by setting $P = P_u \cup P_v$ and $Q = L_\ell \cup L_h$. See Figure 2.

Finally, we consider the general case when G may have multiple connected components. If all components have weight at most $W/2$, a trivial separator exists with $S = \emptyset$. On the other hand, if G has one component of weight at least $W/2$, we can apply the lemma to the largest-weight component and add the remaining components to the smaller of the two subsets V_1, V_2 in terms of weight (which would still have total weight at most $3W/4$). ◀

2.3 Ensuring simplicity of walks

In this subsection, we formulate a type of *legal alternating* closed walks so that a shortest closed walk of this type is automatically a simple cycle. This will be useful later in Section 4. (Warning: both the formulation and the proof of correctness below are somewhat subtle.)

In the following, we imagine that a separator has been computed, and the two parts are colored red and blue, while the separator vertices are colored white.

▶ **Lemma 3.** *Let G be an undirected graph, where each vertex is colored red, blue, or white, and there are no edges between red and blue vertices. Consider a closed walk C which is the concatenation of simple paths $\pi_0^R, \tau_0, \pi_0^B, \tau_0', \ldots, \pi_{k-1}^R, \tau_{k-1}, \pi_{k-1}^B, \tau_{k-1}'$ with $k \geq 1$, where each π_i^R consists of red and white vertices only and has at least one red vertex, each π_i^B consists of blue and white vertices only and has at least one blue vertex, and each τ_i and τ_i' consist of white vertices only (but may be empty). Then C must contain a nontrivial simple cycle.*

Proof. Suppose that two consecutive edges of C are identical, i.e., we see 3 consecutive vertices u, v, u in C. If uv is the last edge of π_i^R and vu is the first edge of τ_i, then u and v are white vertices and we can remove v from C and still have a closed walk satisfying the stated condition. If uv is the last edge of π_i^R, and τ_i is empty, and vu is the first edge of π_i^B, then u and v must again be white vertices and we can remove v from C and still have a closed walk satisfying the stated condition. After handling all other similar cases, we may assume that no two consecutive edges of C are identical. Now, C forms a subgraph where every vertex has degree at least 2, and so must contain a nontrivial simple cycle. ◀

▶ **Lemma 4.** *Let G be an undirected graph, where each vertex is colored red, blue, or white, and there are no edges between red and blue vertices. Let R, B, and W be the subset of red, blue, and white vertices respectively. Define the following:*

- *A simple path π from u to v in G is a* legal red path *if π lies in $G[R \cup W]$, $u, v \in W$, and $\delta_{G[R\cup W]}(u, v) \neq \delta_{G[W]}(u, v)$.*
- *A simple path π from u to v in G is a* legal blue path *if π lies in $G[B \cup W]$, $u, v \in W$, and $\delta_{G[B\cup W]}(u, v) \neq \delta_{G[W]}(u, v)$.*
- *A closed walk C is a* legal alternating walk *if it is the concatenation of simple paths $\pi_0^R, \tau_0, \pi_0^B, \tau_0', \ldots, \pi_{k-1}^R, \tau_{k-1}, \pi_{k-1}^B, \tau_{k-1}'$ with $k \geq 1$, where each π_i^R is a legal red path, each π_i^B is a legal blue path, and each τ_i and τ_i' lie in $G[W]$.*

Then

(i) *the girth of G is at most the length of a shortest legal alternating closed walk;*

(ii) *furthermore, equality holds if the girth of G is not equal of girth of $G[R \cup W]$ or the girth of $G[B \cup W]$.*

Proof. For (i), consider a shortest legal alternating closed walk C. Then C is the concatenation of simple paths $\pi_0^R, \tau_0, \pi_0^B, \tau_0', \ldots, \pi_{k-1}^R, \tau_{k-1}, \pi_{k-1}^B, \tau_{k-1}'$ with $k \geq 1$, where each π_i^R is a legal red path, each π_i^B is a legal blue path, and each τ_i and τ_i' lie in $G[W]$. Each π_i^R must pass through at least one red vertex, because $\delta_{G[R\cup W]}(u(\pi_i^R), v(\pi_i^R)) \neq \delta_{G[W]}(u(\pi_i^R), v(\pi_i^R))$, where $u(\pi)$ and $v(\pi)$ denote the starting and ending vertices of a path π. Similarly, each π_i^B must pass through at least one blue vertex. So, the condition in Lemma 3 is satisfied, and C must contain a nontrivial simple cycle. Hence, the girth of G is at most the length of C.

For (ii), consider a shortest simple cycle C in G, of length g. If there are multiple shortest simple cycles, pick one with the minimum number of non-white vertices. We can express C as a concatenation of simple paths $\pi_0^R, \pi_0^B, \ldots, \pi_{k-1}^R, \pi_{k-1}^B$ with $k \geq 1$, where each π_i^R lies in $G[R \cup W]$, uses at least one red vertex, and starts and ends with white vertices, and each π_i^B lies in $G[B \cup W]$, uses at least one blue vertex, and starts and ends with white vertices. (Note that C contains at least one red and at least one blue vertex, by the assumption in stated in (ii).) If $\delta_{G[R\cup W]}(u(\pi_i^R), v(\pi_i^R)) = \delta_{G[W]}(u(\pi_i^R), v(\pi_i^R))$ for some i, then we can replace π_i^R with a shortest path γ from $u(\pi_i^R)$ to $v(\pi_i^R)$ in $G[W]$, and obtain a closed walk which is

the concatenation of two non-identical simple paths γ and $C \setminus \pi_i^R$, and so must contain a nontrivial simple cycle; the cycle has length at most g and uses fewer non-white vertices: a contradiction. Thus, each π_i^R is a legal red path. Similarly, each π_i^B is a legal blue path. Hence, C is a legal alternating closed walk. ◀

(An astute reader might wonder why we have overcomplicated the definition of legal alternating walk by including the white subpaths τ_i, since part (ii) of the proof of Lemma 4 does not require them. The purpose for including them is computational, as we will see later: without them, intermediate distance matrices may no longer satisfy the bounded-difference property.)

3 Multiple Distances in Clustered Planar Graphs

As a warm-up, we first present an algorithm for a subproblem (of independent interest), on computing all distances with sources and destinations from a given subset X, in an intersection graph of line segments. It is actually more convenient to solve the problem for a more general family of graphs, which we call *clustered planar graphs*, where each vertex in a planar graph is replaced by a "cluster" of c vertices:

▶ **Definition 5.** *A c-*clustered planar graph *refers to the product graph $\widehat{G} = G \times K_c$ where $G = (V, E)$ is a planar graph and K_c is the c-clique. In other words, the vertices of $\widehat{G}$ are pairs $(v, j) \in V \times \{1, \ldots, c\}$, and there is an edge between two vertices (u, i) and (v, j) whenever $uv \in E$ or $u = v$.*

Known shortest-path-related algorithms for planar graphs do not necessarily generalize to clustered planar graphs (because of the lack of Monge property, VC-dimension arguments, etc.). We prove the following result, which beats the obvious $\widetilde{O}(xn)$ time bound, for example, when x is around $\sqrt{n}$:

▶ **Theorem 6.** *Let $\widehat{G} = G \times K_c$ be a c-clustered planar graph, where $G = (V, E)$ has n vertices and $c = O(1)$. Suppose each edge of $\widehat{G}$ has a nonnegative integer weight bounded by $O(1)$. Given a subset X of x vertices in G, we can compute $\delta_{\widehat{G}}[X \times K_c, X \times K_c]$ in $\widetilde{O}(n^{3/2-\alpha} + n^{(1/2+\alpha)(3+\omega)/2} + x^{(3+\omega)/2} + x^2 n^{1/2-\alpha})$ expected time for any given α.*

Proof. We will solve an extended problem: given G, $\widehat{G}$, X, and an additional subset B of β vertices in G such that B has γ components in G, we want to compute $\delta_{\widehat{G}}[(B \cup X) \times K_c, (B \cup X) \times K_c]$. (Initially, $B = \emptyset$ and $\beta = \gamma = 0$.)

Step 1: Build a separator. Give each vertex in $V \setminus B$ weight 1 and each vertex in B weight $1 + w$ for a parameter w (assuming $n + w\beta \gg (1 + w)/\varepsilon$); the total weight is $n + w\beta$. Give each vertex in $V \setminus B$ load 1 and each vertex in B load n/β; the total load is $O(n)$. We apply Lemma 2 for a value p to be set later, to partition V into V_1, V_2, S such that

- $|V_i| + w|V_i \cap B| \le 3(n + w\beta)/4$ for each $i \in \{1, 2\}$;
- there are no edges of G in $V_1 \times V_2$;
- S can be further partitioned into P and Q such that P is the union of at most two paths in G of length $O(p)$, and Q has total load at most $O(n/p)$ – this implies that $|Q| \le O(n/p)$ and $|Q \cap B| \le O(\frac{n/p}{n/\beta}) = O(\beta/p)$ simultaneously.

Let $\widehat{V} = V \times K_c$, $\widehat{S} = S \times K_c$, $\widehat{P} = P \times K_c$, $\widehat{Q} = Q \times K_c$, $\widehat{B} = B \times K_c$ and $\widehat{X} = X \times K_c$.

Step 2: Recurse. For each $i \in \{1,2\}$, we recursively solve the problem for $G_i = G[V_i \cup S]$, $\widehat{G}_i = G_i \times K_c$ (with edge weights inherited from $\widehat{G}$), $B_i = (V_i \cap B) \cup P \cup (Q \cap B)$, and $X_i = V_i \cap X$.

Let $n_i = |V_i \cup S|$, $\beta_i = |B_i|$, γ_i be the number of components of B_i in G_i, and $x_i = |X_i|$. Then $n_1+n_2 \le n+O(p+n/p)$, $\beta_1+\beta_2 \le \beta+O(p+\beta/p)$, $n_i+w\beta_i \le 3(n+w\beta)/4+O(w(p+n/p))$, and $x_1 + x_2 \le x$.

To bound γ_i, consider a component C of B_i in G_i. If C does not visit S, then C is an original component of B (since $S \cap B \subseteq B_i$). If C visits a vertex of P, then C contains one of the two paths in P (since $P \subseteq B_i$). If C visits a vertex of $Q \cap B$, then C can be charged to that vertex. It follows that $\gamma_1 + \gamma_2 \le \gamma + O(1 + \beta/p)$.

Step 3: Compute $\delta_{\widehat{G}}[\widehat{B} \cup \widehat{X}, \widehat{B} \cup \widehat{X}]$. We now compute $\delta_{\widehat{G}}[\widehat{B} \cup \widehat{X}, \widehat{B} \cup \widehat{X}]$ by the following expression:

$$\delta_{\widehat{G}_1}[\widehat{B} \cup \widehat{X}, \widehat{B} \cup \widehat{X}] \vee \delta_{\widehat{G}_2}[\widehat{B} \cup \widehat{X}, \widehat{B} \cup \widehat{X}] \vee \tag{1}$$

$$(\delta_{\widehat{G}}[\widehat{B} \cup \widehat{X}, \widehat{Q}] \star \delta_{\widehat{G}}[\widehat{Q}, \widehat{B} \cup \widehat{X}]) \vee \tag{2}$$

$$\Big((\delta_{\widehat{G}_1}[\widehat{B} \cup \widehat{X}, \widehat{P}] \vee \delta_{\widehat{G}_2}[\widehat{B} \cup \widehat{X}, \widehat{P}]) \star (\delta_{\widehat{G}_1}[\widehat{P}, \widehat{P}] \star \delta_{\widehat{G}_2}[\widehat{P}, \widehat{P}])^n \star (\delta_{\widehat{G}_1}[\widehat{P}, \widehat{B} \cup \widehat{X}] \vee \delta_{\widehat{G}_2}[\widehat{P}, \widehat{B} \cup \widehat{X}])\Big), \tag{3}$$

where $M \vee M'$ denotes the entry-wise minimum of two matrices M and M', and M^k denotes the k-th power under min-plus product.

The expression is correct because for $\widehat{u}, \widehat{v} \in \widehat{B} \cup \widehat{X}$, if a shortest path π from $\widehat{u}$ to $\widehat{v}$ in $\widehat{G}$ avoids $\widehat{S}$, then its length is $\delta_{\widehat{G}_1}(\widehat{u}, \widehat{v})$ or $\delta_{\widehat{G}_2}(\widehat{u}, \widehat{v})$; otherwise, if π passes through $\widehat{Q}$, then its length is $(\delta_{\widehat{G}}[\widehat{B} \cup \widehat{X}, \widehat{Q}] \star \delta_{\widehat{G}}[\widehat{Q}, \widehat{B} \cup \widehat{X}])(\widehat{u}, \widehat{v})$; otherwise, if π passes through $\widehat{P}$, starting from $\widehat{G}_i$ and ending in $\widehat{G}_j$, then its length is $(\delta_{\widehat{G}_1}[\widehat{B} \cup \widehat{X}, \widehat{P}] \star (\delta_{\widehat{G}_1}[\widehat{P}, \widehat{P}] \star \delta_{\widehat{G}_2}[\widehat{P}, \widehat{P}])^n \star \delta_{\widehat{G}_j}[\widehat{P}, \widehat{B} \cup \widehat{X}])(\widehat{u}, \widehat{v})$.

To evaluate the expression, we first compute $\delta_{\widehat{G}}[\widehat{V}, \widehat{Q}]$ and $\delta_{\widehat{G}_i}[\widehat{V}, \widehat{Q}]$ for each $i \in \{1,2\}$ by running BFS from each vertex of $\widehat{Q}$. Since $|Q| = O(n/p)$, this part takes $\widetilde{O}(n^2/p)$ time.

The matrix $\delta_{\widehat{G}_i}[\widehat{B} \cup \widehat{X}, \widehat{B} \cup \widehat{X}]$ is available, since $(B \cup X) \cap (V_i \cup S) \subseteq B_i \cup X_i \cup Q$, and we know $\delta_{\widehat{G}_i}[\widehat{B}_i \cup \widehat{X}_i, \widehat{B}_i \cup \widehat{X}_i]$ from the two recursive calls, and we already know $\delta_{\widehat{G}_i}[\widehat{V}, \widehat{Q}]$.

The matrices $\delta_{\widehat{G}_i}[\widehat{B} \cup \widehat{X}, \widehat{P}]$ and $\delta_{\widehat{G}_i}[\widehat{P}, \widehat{P}]$ are also available, since $P \subseteq B_i$.

We now describe how to compute $\delta_{\widehat{G}}[\widehat{B} \cup \widehat{X}, \widehat{Q}] \star \delta_{\widehat{G}}[\widehat{Q}, \widehat{B} \cup \widehat{X}]$ in (2). For any component $\widehat{C} \subseteq \widehat{B}$, the columns of $\delta_{\widehat{G}}[\widehat{C}, \widehat{Q}]$ satisfy the bounded-difference property, after reordering the rows according to an Euler tour of a spanning tree of C – some rows may need to be duplicated, but the number of rows increases by only a constant factor. (Note that all vertices in a common cluster $\{v\} \times K_c$ are within a constant distance from each other.) Since B has γ components, we can compute $\delta_{\widehat{G}}[\widehat{B}, \widehat{Q}] \star \delta_{\widehat{G}}[\widehat{Q}, \widehat{B} \cup \widehat{X}]$ by Lemma 1 in $\widetilde{O}(\gamma(\beta + n/p + x)^{(3+\omega)/2})$ time. The product $\delta_{\widehat{G}}[\widehat{B} \cup \widehat{X}, \widehat{Q}] \star \delta_{\widehat{G}}[\widehat{Q}, \widehat{B}]$ is symmetric. It remains to compute $\delta_{\widehat{G}}[\widehat{X}, \widehat{Q}] \star \delta_{\widehat{G}}[\widehat{Q}, \widehat{X}]$: here, since we don't have the bounded-difference property, we naively compute the product in $O(|X|^2|Q|) = O(x^2 n/p)$ time.

Finally, we describe how to compute (3). The columns of $\delta_{\widehat{G}_i}[\widehat{P}, \widehat{P}]$ satisfy the bounded-difference property, after dividing the matrix into two parts and reordering the rows, since P is the union of at most two walks; so, we can compute $\delta_{\widehat{G}_1}[\widehat{P}, \widehat{P}] \star \delta_{\widehat{G}_2}[\widehat{P}, \widehat{P}]$ in $\widetilde{O}(p^{(3+\omega)/2})$ time. We can compute the n-th power by repeated squaring using $O(\log n)$ bounded-difference matrix products. (Note that if the columns of a matrix M satisfy the bounded-difference property, then the columns of a product $M \star M'$ also satisfy the bounded-difference property.)

The rows of $\delta_{\widehat{G}_i}[\widehat{B} \cup \widehat{X}, \widehat{P}]$ satisfy the bounded-difference property (after dividing the matrix into two parts), and so the other products can be computed in a similar way.

Running time. The time complexity $T(n, \beta, \gamma, x)$ of the algorithm satisfies the following recurrence:

$$T(n,\beta,\gamma,x) \le \max_{\substack{n_1,n_2,\beta_1,\beta_2,x_1,x_2:\\ n_i+w\beta_i\le 3(n+w\beta)/4+O(w(p+n/p)),\\ n_1+n_2\le n+O(p+n/p),\\ \beta_1+\beta_2\le\beta+O(p+\beta/p),\\ \gamma_1+\gamma_2\le\gamma+O(1+\beta/p),\\ x_1+x_2\le x}} \Big(T(n_1,\beta_1,\gamma_1,x_1)+T(n_2,\beta_2,\gamma_2,x_2)+ \widetilde{O}(n+n^2/p+ \gamma(\beta+p+n/p+x)^{(3+\omega)/2}+x^2n/p)\Big).$$

Solving the recurrence. Intuitively, one could choose $p \approx n^{1/2+\alpha}$, and think of β as $\widetilde{O}(p)$ and γ as $\widetilde{O}(1)$, and see that the cost is upper bounded by $n^{3/2-\alpha} + x^{(3+\omega)/2} + x^2 n^{1/2-\alpha}$ for any sufficiently small positive α (as long as $\omega < 3$), at least at the top few levels of the recursion.

Formally, let r be a fixed parameter, to be set later, with $r \gg w(p + n/p)\log n$. For a node u in the recursion tree, let $n_u, \beta_u, \gamma_u, X_u$ denote its input parameters. Suppose we stop the recursion at a node u whenever $n_u + w\beta_u \le r$; in particular, this condition implies $n_u \le r$ and $\beta_u \le r/w$ simultaneously. Then the parent v of each leaf satisfies $\mu_v + w\beta_v > r$. This recursion subtree has depth at most $O(\log \frac{n+w\beta}{r})$. Let ξ be the size of this recursion subtree. Then $\sum_v (n_v + w\beta_v) \le O((n+w\beta+\xi\cdot w(p+n/p))\log\frac{n+w\beta}{r})$, where the summation is over each v that is the parent of some leaf. Thus, $\xi \le O(\frac{1}{r}\sum_v(n_v+w\beta_v)) \le O(\frac{n+w\beta}{r}\log\frac{n+w\beta}{r}) + o(\xi)$, implying that $\xi \le O(\frac{n+w\beta}{r}\log\frac{n+w\beta}{r})$.

This leads to a new recurrence:

$$T(n,\beta,\gamma,x) \le \max_{\substack{x_1,x_2,\ldots:\\ \sum_k x_k\le x}} \sum_{k=1}^{O(\frac{n+w\beta}{r}\log\frac{n+w\beta}{r})} \big(T(r, r/w, \gamma+\widetilde{O}(1+\beta/p), x_k)+ \widetilde{O}\big(n+n^2/p+ \gamma(\beta+p+n/p+x)^{(3+\omega)/2}+x^2n/p\big)\big).$$

We now set $\beta = n^{1/2+\alpha}$, $w = n^{1/2-\alpha}$, $r = n/t$, and $p = n^{1/2+\alpha}/\log^{1+\varepsilon} n$ for a sufficiently large constant t. Note that indeed $r \gg w(p + n/p)\log n$. By loosely bounding r/w by $(n/t)^{1/2+\alpha}$, the recurrence becomes:

$$T(n,n^{1/2+\alpha},\gamma,x) \le \max_{\substack{x_1,x_2,\ldots:\\ \sum_k x_k\le x}} \sum_{k=1}^{O(t\log t)} \big(T(n/t,(n/t)^{1/2+\alpha},\gamma+\widetilde{O}(1),x_k)+ \widetilde{O}(n^{3/2-\alpha}+\gamma(n^{(1/2+\alpha)(3+\omega)/2}+x^{(3+\omega)/2}) x^2n^{1/2-\alpha})\big).$$

This solves to $T(n, n^{1/2+\alpha}, \gamma, x) = \widetilde{O}(n^{3/2-\alpha}+(1+\gamma)(n^{(1/2+\alpha)(3+\omega)/2}+x^{(3+\omega)/2})+x^2n^{1/2-\alpha})$. ◀

▶ **Corollary 7.** *Given a set A of line segments in $\mathbb{R}^2$ and a subset $X \subset A$ of x segments, we can compute $\delta_G[X, X]$, where G is the intersection graph of A, in $\widetilde{O}(\mu^{3/2-\alpha}+\mu^{(1/2+\alpha)(3+\omega)/2}+ x^{(3+\omega)/2} + x^2\mu^{1/2-\alpha})$ expected time for any given α, where μ is the number of endpoints and intersection points of A.*

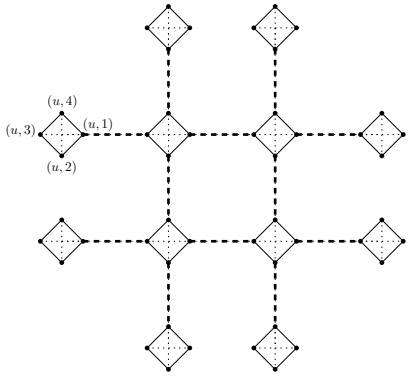


Figure 3 The clustered planar graph $\widehat{H}$. The dotted edges and the dashed edges have weight 0, while the solid edges have weight 1. Edges with weight 2 are omitted.

Proof. For simplicity, we assume that the given segments are in general position (e.g., no 3 segments intersect at a common point).

First we compute the intersections of the line segments in A, for example, by applying Bentley and Ottmann's plane-sweep algorithm [4], which runs in $O(\mu \log \mu)$ time.

Let H be the *planarized intersection graph* of A, i.e., the planar graph formed by the arrangement of A, where the vertices consist of the segment endpoints and intersection points of A, and the edges of H correspond to pairs of adjacent vertices in the arrangement.

Let $\widehat{H}$ be the clustered planar graph $H \times K_4$ (see Figure 3). Number the edges incident to each vertex of H from $\{1, 2, 3, 4\}$ in clockwise order. Give the edge $(u, i)(v, j)$ weight 0 in $\widehat{H}$ if uv is in H and the i-th edge incident to u coincides with the j-th edge incident to v. In addition, for each vertex u of degree 4 in H, give the edges $(u, 1)(u, 2)$, $(u, 2)(u, 3)$, $(u, 3)(u, 4)$, $(u, 4)(u, 1)$ weight 1, and give the edges $(u, 1)(u, 3)$ and $(u, 2)(u, 4)$ weight 0. Give all other edges weight 2 in $\widehat{H}$. Let $u(s)$ denote the left endpoint of a segment s. Then for any two segments s and t, the distance between s and t in G is equal to the distance between $(u(s), 1)$ and $(u(t), 1)$ in $\widehat{H}$. So, the result immediately follows by applying Theorem 6 to $\widehat{H}$ and $\{u(s) : s \in X\}$. ◀

4 Girth for Line Segments

We now present our girth algorithm for intersection graphs of line segments. It is tempting to continue to work within the clustered planar graph framework, but unfortunately a clustered planar graph by definition contains short cycles in every cluster (furthermore, when generalizing from segments to curves later, an additional difficulty is that simple cycles in the planarized intersection graph may not correspond to simple cycles in the intersection graph – see Figure 4).

We propose another recursive algorithm, which is also based on our separator lemma but proceeds differently: on one hand, we have to work with intersection graphs instead of clustered planar graphs, and also have to deal with the issue of ensuring simplicity of walks; on the other hand, Corollary 7 is available as a subroutine to compute distances, so fewer input arguments are needed in the new recursive algorithm (notably, we don't need an additional input set B).



Figure 4 These are not simple cycles in the intersection graph!

▶ **Theorem 8.** *Given a set A of n line segments in $\mathbb{R}^2$, we can compute the girth g_G of its intersection graph G of A in $O(n^{1.483})$ expected time, under the promise that the girth is more than 4.*

Proof. It is known that any intersection graph of n line segments with girth more than 4 must have $O(n)$ edges [42].

Step 1: Build a separator. We first construct the planarized intersection graph $H = (V_H, E_H)$, as in the proof of Corollary 7, in $\widetilde{O}(n)$ time. Note that $\mu = |V_H| = O(n)$. Define the weight of a point $p \in V_H$ to be 1 if it is the left endpoint of a segment, and 0 otherwise; the total weight is n. We can apply Lemma 2 (with unit loads), for a value p to be set later, to partition V_H into V_1, V_2, S_H such that

- V_i has weight at most $3n/4$ for each $i \in \{1, 2\}$;
- there are no edges of H in $V_1 \times V_2$;
- S_H can be further partitioned into P_H and Q_H such that P_H is the union of at most two paths in H of length $O(p)$, and $|Q_H| = O(n/p)$.

For each $i \in \{1, 2\}$, let A_i consist of the segments $s \in A$ for which all points of V_H on s are in V_i. Note that $|A_i| \le 3n/4$. Let P consist of the segments with at least one point in P_H. Let Q consist of the segments with at least one point in Q_H and no points in P_H. Let $S = P \cup Q$. Note that P is the union of at most two walks of length $O(p)$ in G (since any path in H corresponds to a walk in G).

Step 2: Recurse. For each $i \in \{1, 2\}$, we recursively solve the problem for $A_i \cup P$ to obtain the girth g_{G_i} of the intersection graph G_i of $A_i \cup P$. (Note that the assumption that G has girth more than 4 implies that G_i also has girth more than 4.)

Let $n_i = |A_i \cup P|$. Then $n_1, n_2 \le 3n/4 + O(p)$ and $n_1 + n_2 \le n + O(p)$.

Step 3: Compute $\delta_{G_i}[P, P]$. For each $i \in \{1, 2\}$, we compute $\delta_{G_i}[P, P]$ by Corollary 7 in $\widetilde{O}(n^{3/2-\alpha} + n^{(1/2+\alpha)(3+\omega)/2} + p^{(3+\omega)/2} + p^2 n^{1/2-\alpha})$ expected time.

Step 4: Compute the girth g_G. We first run BFS from each segment of Q to find the length g_Q of the shortest cycle visiting Q in G. This part takes $\widetilde{O}(n^2/p)$ time.

Next, we compute the length g_{legal} of the shortest legal alternating closed walk where the segments in A_1 are red, the segments in A_2 are blue, and the segments in P are white (discarding the segments in Q). By Lemma 4, the girth $g_{G[A_1 \cup A_2 \cup P]}$ of $G[A_1 \cup A_2 \cup P]$ is equal to $\min\{g_{\text{legal}}, g_{G_1}, g_{G_2}\}$. The final answer is $g_G = \min\{g_{G[A_1 \cup A_2 \cup P]}, g_Q\}$.

To compute g_{legal}, for each $i \in \{1, 2\}$, we first compute the "legal" distance matrices $\delta'_{G_i}[P, P]$: for $u, v \in P$, let

$$\delta'_{G_i}(u, v) = \begin{cases} \delta_{G_i}(u, v) & \text{if } \delta_{G_i}(u, v) \neq \delta_{G[P]}(u, v), \\ +\infty & \text{otherwise.} \end{cases}$$

We already know the matrix $\delta_{G_i}[P, P]$. We can compute $\delta_{G[P]}$ naively in $\widetilde{O}(p^\omega)$ time. Consequently, we know the matrix $\delta'_{G_i}[P, P]$.

Next, we compute

$$\delta''_{G_i}[P,P] = \delta'_{G_i}[P,P] \star \delta_{G[P]}.$$

Although $\delta'_{G_i}[P,P]$ may not satisfy a bounded-difference property, the rows of $\delta_{G[P]}$ satisfy the bounded-difference property, after dividing the matrix into two parts and reordering the columns, since P is the union of at most two walks. So, we can compute $\delta''_{G_i}[P,P]$ in $\widetilde{O}(p^{(3+\omega)/2})$ time by Lemma 1. It also follows that the rows of $\delta''_{G_i}[P,P]$ satisfy the bounded-difference property. Finally, let

$$Z_k = \bigvee_{i=1}^{k} (\delta''_{G_1}[P,P] \star \delta''_{G_2}[P,P])^i.$$

From the definition of legal alternating walks, we see that $Z_n(u,u)$ is precisely the length of the shortest legal alternating closed walk starting at $u \in P$. Thus, g_{legal} is the smallest diagonal entry in Z_n. We can compute Z_n recursively by the following formula:

$$Z_k = (Z_{k/2} \star Z_{k/2}) \vee (\delta''_{G_1}[P,P] \star \delta''_{G_2}[P,P]).$$

(This assumes that k is even, and the odd case is similar.) This requires $O(\log n)$ min-plus matrix products, and all intermediate matrices satisfy the bounded-difference property. So, Z_n can be computed in $\widetilde{O}(p^{(3+\omega)/2})$ time.

Running time. The time complexity $T(n)$ of the algorithm satisfies the following recurrence:

$$T(n) \le \max_{\substack{n_1,n_2:\\ n_1,n_2\le 3n/4+O(p),\\ n_1+n_2\le n+O(p)}} \Big(T(n_1)+T(n_2)+\widetilde{O}(n^2/p + n^{3/2-\alpha} + n^{(1/2+\alpha)(3+\omega)/2} + p^{(3+\omega)/2} + p^2 n^{1/2-\alpha})\Big).$$

To balance the three terms n^2/p, $n^{(1/2+\alpha)(3+\omega)/2}$, and $p^2 n^{1/2-\alpha}$, we set $p = n^{1/2+\rho}$ and $\alpha = 3\rho$, with $\rho = \frac{3-\omega}{22+6\omega} > 0.0173$, using the latest bound $\omega < 2.371339$ [2]. Then the recurrence becomes:

$$T(n) \le \max_{\substack{n_1,n_2:\ n_1,n_2\le 3n/4+o(n),\\ n_1+n_2\le n+o(n)}} \big(T(n_1)+T(n_2)+\widetilde{O}(n^{3/2-\rho})\big),$$

which solves to $T(n) = \widetilde{O}(n^{3/2-\rho}) \le O(n^{1.483})$. ◀

Finally, we remove the assumption that girth is more than 4:

▶ **Corollary 9.** *Given n line segments in $\mathbb{R}^2$, we can compute the girth of the intersection graph in $O(n^{1.483})$ expected time.*

Proof. We can check whether the intersection graph G contains a 3-cycle in $O(n^{1.408})$ time by an algorithm of Chan [14, Theorem 5.1], or a 4-cycle in $\widetilde{O}(n)$ time by another algorithm of Chan [14, Theorem 6.2]. So, we may assume that the girth is greater than 4, and the result follows from Theorem 8. ◀

In the full version, we describe how to modify our algorithms more generally for arbitrary algebraic curves or semialgebraic sets that are connected and have constant description complexity.

Our results are unlikely to be optimal. Although we do not know how to improve Corollary 7, we are currently working on ideas that might further improve the time bound of Corollary 9 for the case of line segments, by exploiting special properties about intersection graphs with large girth.

References

1 Pankaj K. Agarwal and Marc J. van Kreveld. Connected component and simple polygon intersection searching. *Algorithmica*, 15(6):626–660, 1996. doi:10.1007/BF01940884.

2 Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proc. 36th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2039, 2025. doi:10.1137/1.9781611978322.63.

3 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

4 Jon Louis Bentley and Thomas Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Computers*, 28(9):643–647, 1979. doi:10.1109/TC.1979.1675432.

5 Béla Bollobás. Chromatic number, girth and maximal degree. *Discret. Math.*, 24(3):311–314, 1978. doi:10.1016/0012-365X(78)90102-4.

6 Édouard Bonnet, Sergio Cabello, and Wolfgang Mulzer. Maximum matchings in geometric intersection graphs. *Discret. Comput. Geom.*, 70(3):550–579, 2023. doi:10.1007/S00454-023-00564-3.

7 Karl Bringmann and Alejandro Cassis. Faster knapsack algorithms via bounded monotone min-plus-convolution. In *Proc. 49th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 229 of *LIPIcs*, pages 31:1–31:21, 2022. doi:10.4230/LIPIcs.ICALP.2022.31.

8 Karl Bringmann, Anita Dürr, and Adam Polak. Even faster knapsack via rectangular monotone min-plus convolution and balancing. In *Proc. 32nd European Symposium on Algorithms (ESA)*, volume 308 of *LIPIcs*, pages 33:1–33:15, 2024. doi:10.4230/LIPIcs.ESA.2024.33.

9 Karl Bringmann, Fabrizio Grandoni, Barna Saha, and Virginia Vassilevska Williams. Truly subcubic algorithms for language edit distance and RNA folding via fast bounded-difference min-plus product. *SIAM J. Comput.*, 48(2):481–512, 2019. doi:10.1137/17M112720X.

10 Karl Bringmann, Sándor Kisfaludi-Bak, Marvin Künnemann, André Nusser, and Zahra Parsaeian. Towards sub-quadratic diameter computation in geometric intersection graphs. In *Proc. 38th International Symposium on Computational Geometry (SoCG)*, volume 224 of *LIPIcs*, pages 21:1–21:16, 2022. doi:10.4230/LIPIcs.SOCG.2022.21.

11 Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Trans. Algorithms*, 15(2):21:1–21:38, 2019. doi:10.1145/3218821.

12 Sergio Cabello and Wolfgang Mulzer. Minimum cuts in geometric intersection graphs. *Comput. Geom.*, 94:101720, 2021. doi:10.1016/J.COMGEO.2020.101720.

13 Parinya Chalermsook, Jittat Fakcharoenphol, and Danupon Nanongkai. A deterministic near-linear time algorithm for finding minimum cuts in planar graphs. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 828–829, 2004. doi:10.5555/982792.982916.

14 Timothy M. Chan. Finding triangles and other small subgraphs in geometric intersection graphs. In *Proc. 34th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1777–1805, 2023. doi:10.1137/1.9781611977554.CH68.

15 Timothy M. Chan, Hsien-Chih Chang, Jie Gao, Sándor Kisfaludi-Bak, Hung Le, and Da Wei Zheng. Truly subquadratic time algorithms for diameter and related problems in graphs of bounded VC-dimension. *CoRR*, abs/2510.16346, 2025. To appear in FOCS 2025. doi:10.48550/arXiv.2510.16346.

16 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proc. 47th ACM Symposium on Theory of Computing (STOC)*, pages 31–40, 2015. doi:10.1145/2746539.2746568.

17 Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in geometric intersection graphs. *J. Comput. Geom.*, 10(1):27–41, 2019. doi:10.20382/JOCG.V10I1A2.

18 L. Sunil Chandran and C. R. Subramanian. Girth and treewidth. *J. Comb. Theory B*, 93(1):23–32, 2005. doi:10.1016/J.JCTB.2004.05.004.

19 Hsien-Chih Chang and Hsueh-I Lu. Computing the girth of a planar graph in linear time. *SIAM J. Comput.*, 42(3):1077–1094, 2013. doi:10.1137/110832033.

20 Shucheng Chi, Ran Duan, and Tianle Xie. Faster algorithms for bounded-difference min-plus product. In *Proc. 33th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1435–1447, 2022. doi:10.1137/1.9781611977073.60.

21 Shucheng Chi, Ran Duan, Tianle Xie, and Tianyi Zhang. Faster min-plus product for monotone instances. In *Proc. 54th ACM Symposium on Theory of Computing (STOC)*, pages 1529–1542, 2022. doi:10.1145/3519935.3520057.

22 R. J. Cook. Chromatic number and girth. *Periodica Mathematica Hungarica*, 6(1):103–107, 1975. doi:10.1007/BF02018401.

23 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate texts in mathematics*. Springer, 2015. doi:10.1007/978-3-662-70107-2.

24 Hristo N. Djidjev. Computing the girth of a planar graph. In *Proc. 27th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1853 of *LIPIcs*, pages 821–831, 2000. doi:10.1007/3-540-45022-X_69.

25 Hristo N. Djidjev. A faster algorithm for computing the girth of planar and bounded genus graphs. *ACM Trans. Algorithms*, 7(1):3:1–3:16, 2010. doi:10.1145/1868237.1868240.

26 Guillaume Ducoffe, Michel Habib, and Laurent Viennot. Diameter, eccentricities and distance oracle computations on H-minor free graphs and graphs of bounded (distance) Vapnik-Chervonenkis dimension. *SIAM J. Comput.*, 51(5):1506–1534, 2022. doi:10.1137/20M136551X.

27 Paul Erdös. Graph theory and probability. *Canadian Journal of Mathematics*, 11:34–38, 1959. doi:10.1007/978-0-8176-4842-8_19.

28 Jeff Erickson. Planar separators. Notes, 2023. URL: https://jeffe.cs.illinois.edu/teaching/comptop/2023/notes/14-planar-separators.pdf.

29 Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.*, 72(5):868–889, 2006. doi:10.1016/J.JCSS.2005.05.007.

30 Pawel Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{O}(n^{5/3})$ time. *SIAM J. Comput.*, 50(2):509–554, 2021. doi:10.1137/18M1193402.

31 Yuzhou Gu, Adam Polak, Virginia Vassilevska Williams, and Yinzhan Xu. Faster monotone min-plus product, range mode, and single source replacement paths. In *Proc. 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 198 of *LIPIcs*, pages 75:1–75:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.75.

32 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978. doi:10.1137/0207033.

33 Haim Kaplan, Katharina Klost, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Triangles and girth in disk graphs and transmission graphs. In *Proc. 27th European Symposium on Algorithms (ESA)*, volume 144 of *LIPIcs*, pages 64:1–64:14, 2019. doi:10.4230/LIPIcs.ESA.2019.64.

34 Philip Klein and Shay Mozes. Optimization algorithms for planar graphs. Book draft, 2024. URL: https://planarity.org.

35 Katharina Klost and Wolfgang Mulzer. Robust algorithms for finding triangles and computing the girth in unit disk and transmission graphs. *CoRR*, abs/2405.01180, 2024. doi:10.48550/arXiv.2405.01180.

36 Hung Le and Christian Wulff-Nilsen. VC set systems in minor-free (di)graphs and applications. In *Proc. 35th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5332–5360, 2024. doi:10.1137/1.9781611977912.192.

37 Jason Li and Merav Parter. Planar diameter via metric compression. In *Proc. 51st ACM Symposium on Theory of Computing (STOC)*, pages 152–163, 2019. doi:10.1145/3313276.3316358.

38 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, 36(2):177–189, 1979. doi:10.1137/0136016.

39 László Lovász. On chromatic number of finite set-systems. *Acta Mathematica Academiae Scientiarum Hungarica*, 19(1):59–67, 1968. doi:10.1007/BF01894680.

40 Xiao Mao. Breaking the cubic barrier for (unweighted) tree edit distance. In *Proc. 62nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 792–803, 2021. doi:10.1109/FOCS52979.2021.00082.

41 Gary L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *J. Comput. Syst. Sci.*, 32(3):265–279, 1986. doi:10.1016/0022-0000(86)90030-9.

42 János Pach and Micha Sharir. On planar intersection graphs with forbidden subgraphs. *J. Graph Theory*, 59(3):205–214, 2008. doi:10.1002/JGT.20332.

43 Barna Saha and Christopher Ye. Faster approximate all pairs shortest paths. In *Proc. 35th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4758–4827, 2024. doi:10.1137/1.9781611977912.170.

44 Jeffrey D. Ullman and Mihalis Yannakakis. High-probability parallel transitive-closure algorithms. *SIAM J. Comput.*, 20(1):100–125, 1991. doi:10.1137/0220006.

45 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proc. International Congress of Mathematicians (ICM)*, pages 3447–3487, 2018. doi:10.1142/9789813272880_0188.

46 Oren Weimann and Raphael Yuster. Computing the girth of a planar graph in $O(n \log n)$ time. *SIAM J. Discret. Math.*, 24(2):609–616, 2010. doi:10.1137/090767868.

47 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002. doi:10.1145/567112.567114.

Linear-Time $(1+\varepsilon)$-Approximation Algorithms for Two-Line-Center Problems

Chaeyoon Chung
Department of Computer Science and Engineering, Pohang University of Science and Technology, South Korea

Anil Maheshwari
School of Computer Science, Carleton University, Ottawa, Canada

Michiel Smid
School of Computer Science, Carleton University, Ottawa, Canada

Abstract

Given a set S of n points in the plane, we study the two-line-center problem: finding two lines that minimize the maximum distance from each point in S to its closest line. We present a $(1+\varepsilon)$-approximation algorithm for the two-line-center problem that runs in $O((n/\varepsilon)\log(1/\varepsilon))$ time, which improves the previously best $O(n\log n + (n/\varepsilon^2)\log(1/\varepsilon) + (1/\varepsilon^3)\log(1/\varepsilon))$-time algorithm. We also consider three variants of this problem, in which the orientations of the two lines are restricted: (1) the orientation of one of the two lines is fixed, (2) the orientations of both lines are fixed, and (3) the two lines are required to be parallel. For each of these three variants, we give the first $(1+\varepsilon)$-approximation algorithm that runs in linear time. In particular, for the variant where the orientation of one of the two lines is fixed, we also give an improved exact algorithm that runs in $O(n\log n)$ time and show that it is optimal.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Approximation algorithm, two-line-center problem, k-line-center problem, projective clustering, ε-certificate, ε-coreset, width of a point set

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.31

Related Version *Full Version*: `https://arxiv.org/pdf/2601.03516`

Funding Chaeyoon Chung was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(RS-2025-25436016). Anil Maheshwari and Michiel Smid were supported by NSERC.

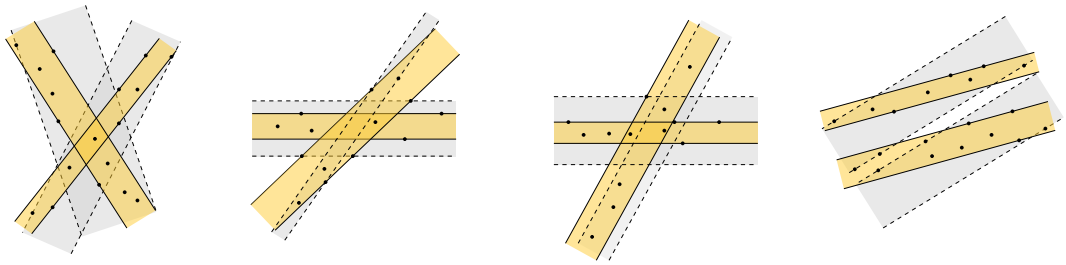
Acknowledgements This work was done while Chaeyoon Chung was visiting Carleton University.

1 Introduction

Given a set S of n points in the plane, we study the two-line-center problem: finding two lines that minimize the maximum distance from each point in S to its closest line. This is a special case of the k-line-center problem, which asks, for a given set S of n points in the plane, to find k lines minimizing the maximum distance from each point in S to its closest line. A more general framework is projective clustering: Given $P \subset \mathbb{R}^d$, the goal is to find k flats of dimension j that best fit P under a given distance measure. It has wide applications in data mining [4, 5], unsupervised learning [23], database management [10], and computer vision [24].

The k-line-center problem is known to be NP-hard [21] when k is part of the input (even in the plane), and various approaches including approximation algorithms and randomized methods [1, 3, 7, 9, 11, 13, 16, 17, 18] have been studied.

© Chaeyoon Chung, Anil Maheshwari, and Michiel Smid;
licensed under Creative Commons License CC-BY 4.0



(a) General (b) One fixed orientation (c) Two fixed orientations (d) Parallel

Figure 1 In each case, both yellow and gray pairs satisfy the restriction, but the yellow pair has the smaller maximum width.

When $k = 1$, the problem becomes computing the width of a point set. This can be done exactly in $O(n \log n)$ time [22], and an $\Omega(n \log n)$ lower bound is known [20]. A $(1+\varepsilon)$-approximation algorithm running in $O(n + 1/\varepsilon)$ time was given in [11].

When $k = 2$, i.e., the two-line-center problem, an exact algorithm with running time $O(n^2 \log^2 n)$ is known [19]. The authors in [2] presented a $(1+\varepsilon)$-approximation algorithm that runs in $O(n \log n + (n/\varepsilon^2) \log(1/\varepsilon) + \varepsilon^{-7/2} \log(1/\varepsilon))$ time. Later, as the $(1+\varepsilon)$-approximation algorithm for the width problem was improved from $O(n + \varepsilon^{-3/2})$ to $O(n + 1/\varepsilon)$ [11], their algorithm is consequently improved to $O(n \log n + (n/\varepsilon^2) \log(1/\varepsilon) + (1/\varepsilon^3) \log(1/\varepsilon))$, even though it is not explicitly mentioned.

Observe that the two-line-center problem is equivalent to finding a pair of slabs whose union covers the input point set while minimizing their maximum width.

We study the two-line-center problem and three natural variants. Let $\mu = \min\{n, 1/\varepsilon^2\}$, and let w^* denote the maximum width of an optimal pair of slabs in each problem. Each variant is illustrated in Figure 1. Table 1 summarizes the previous and new results.

1. **General:** Given a set S of n points in the plane, we study the problem of finding a pair of slabs whose union covers S while minimizing the maximum of their widths. We present an $O(n + (\mu/\varepsilon) \log \mu)$-time algorithm that finds a pair of slabs whose union covers S and whose maximum width is at most $(1+\varepsilon)w^*$. The running time can be expressed as both $O((n/\varepsilon) \log(1/\varepsilon))$ and $O(n + (1/\varepsilon^3) \log(1/\varepsilon))$.
2. **One Fixed Orientation:** Given a set S of n points in the plane and an orientation θ, we study the problem of finding a pair of slabs such that one of them has orientation θ and their union covers S while minimizing the maximum of their widths. We present an exact algorithm that runs in $O(n \log n)$ time, improving the previously best $O(n \log^3 n)$-time algorithm [6], and show that this is optimal.

   We also present an $O(n + \mu \log \mu)$-time algorithm that finds a pair of slabs such that one of them has orientation θ, their union covers S, and their maximum width is at most $(1+\varepsilon)w^*$. The running time can be expressed as $O(n \log(1/\varepsilon))$ and $O(n + 1/\varepsilon^2 \log(1/\varepsilon))$.
3. **Two Fixed Orientations:** Given a set S of n points in the plane and two orientations, θ_1 and θ_2, we study the problem of finding two slabs, one has orientation θ_1 and the other has orientation θ_2, whose union covers S while minimizing the maximum of their widths. We present an $O(n + 1/\varepsilon)$-time algorithm that finds a pair of slabs such that one has orientation θ_1 and the other has orientation θ_2, their union covers S, and their maximum width is at most $(1+\varepsilon)w^*$.
4. **Parallel:** Given a set S of n points in the plane, we study the problem of finding a pair of parallel slabs whose union covers S while minimizing the maximum of their widths. We present an $O(n + \mu/\varepsilon)$-time algorithm that finds a pair of parallel slabs such that their union covers S and their maximum width is at most $(1+\varepsilon)w^*$.

Table 1 A summary of our results and previous work.

	Exact	Approximation
General	$O(n^2 \log^2 n)$ [19]	$O(n \log n + \frac{n}{\varepsilon^2} \log \frac{1}{\varepsilon} + \frac{1}{\varepsilon^3} \log \frac{1}{\varepsilon})$ [2, 11]
		$\boldsymbol{O(\frac{n}{\varepsilon} \log \frac{1}{\varepsilon})}$, $\boldsymbol{O(n + \frac{1}{\varepsilon^3} \log \frac{1}{\varepsilon})}$ (Sec. 5)
One fixed orientation	$O(n \log^3 n)$ [6]	-
	$\boldsymbol{O(n \log n)}$ (Sec. 4)	$\boldsymbol{O(n \log \frac{1}{\varepsilon})}$, $\boldsymbol{O(n + \frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})}$ (Sec. 4)
Two fixed orientations	$O(n \log n)$ [6]	-
		$\boldsymbol{O(n + \frac{1}{\varepsilon})}$ (full version)
Parallel	$O(n^2)$ [8]	-
		$\boldsymbol{O(\frac{n}{\varepsilon})}$, $\boldsymbol{O(n + \frac{1}{\varepsilon^3})}$ (full version)

1.1 Outline

In Section 2, we introduce the notation that will be used throughout the paper.

In Section 3, we introduce three key ingredients used throughout the paper. In Section 3.1, we modify the concept of an *anchor pair* (as introduced in [2]) and show how to compute it. In Section 3.2, we use an anchor pair to compute a 10-approximation for the general two-line-center problem in $O(n)$ time. In Section 3.3, we recall the concept of an *ε-certificate* (as defined in [3]), a subset of the input point set that suffices to obtain a $(1+\varepsilon)$-approximate solution, and show how to compute it in $O(n)$ time using an anchor pair from Section 3.1 and a 10-approximate solution from Section 3.2.

In Section 4, we address the one-fixed-orientation two-line-center problem. In Section 4.2, we give an $O(n)$-time decision algorithm, assuming the points are sorted. In Section 4.3, we present an $O(n \log n)$-time exact algorithm and prove its optimality. In Section 4.4, we give a $(1+\varepsilon)$-approximation algorithm.

In Section 5, we address the general two-line-center problem. Using an anchor pair (Section 3.1) and a 10-approximate solution (Section 3.2), we compute a candidate set of orientations for one of the slabs of a $(1+\varepsilon)$-approximate solution. Combining this with the algorithm from Section 4 and an ε-certificate (Section 3.3), we obtain a $(1+\varepsilon)$-approximation algorithm.

Due to space constraints, we address the remaining two problems in the full version.

For the two-fixed-orientations two-line-center problem, we first present a 2-approximation algorithm, and then refine it to obtain a $(1+\varepsilon)$-approximation.

For the parallel two-line-center problem, we use the *gap-ratio* of a pair of slabs (as defined in [12]) and consider two cases separately, depending on whether the gap-ratio of an optimal pair is small or large. Combining these with an ε-certificate, we obtain a $(1+\varepsilon)$-approximation algorithm.

2 Preliminaries

For a set P of points in the plane, we denote its diameter by $\mathrm{diam}(P)$, its width by $\mathrm{width}(P)$, and its convex hull by $\mathrm{conv}(P)$. If $P = \emptyset$, then $\mathrm{width}(P) = 0$. For a slab σ, we denote its width by $w(\sigma)$, which is the distance between the two bounding lines of σ (see Figure 2(a)).

For a line ℓ, we say that ℓ has orientation $\theta \in [0, \pi)$ if θ is the counterclockwise angle from the x-axis to ℓ. A slab σ is said to have orientation $\theta \in [0, \pi)$ if its two bounding lines have orientation θ (see Figure 2(a)). For a finite set P of points in the plane, let $\sigma_\theta(P)$ denote the minimum-width slab of orientation θ that encloses P, and define $\mathrm{width}_\theta(P) = w(\sigma_\theta(P))$ (see Figure 2(b)).

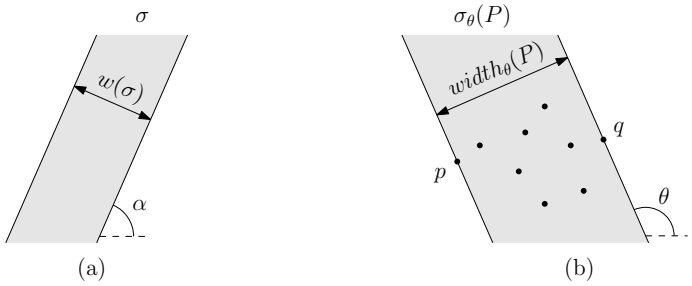


Figure 2 (a) The orientation of σ is α. (b) For a set P of points and an orientation θ, $\sigma_\theta(P)$ and $\mathrm{width}_\theta(P)$ are shown. Observe that (p, q) is an antipodal pair of P with respect to θ.

For a set P of points in the plane and an orientation $\theta \in [0, \pi)$, we say that a pair (p, q) with $p, q \in P$ is an *antipodal pair* of P with respect to θ if p lies on one bounding line of $\sigma_\theta(P)$ and q lies on the other (see Figure 2(b)).

For two finite sets A and B in the plane, let $d(A, B)$ denote the minimum distance between a point of A and a point of B. For two points p and q in the plane, let ℓ_{pq} be the line passing through p and q.

For each of the four problems, we assume that $w^* > 0$, where w^* denotes the maximum width of an optimal pair of slabs for that problem. The case where $w^* = 0$ can be detected and handled in linear time. See the appendix in the full version.

Throughout the paper, let $\varepsilon > 0$ be a sufficiently small real number and let S denote an input set of n points in the plane. Due to space constraints, the proofs of the theorems and lemmas marked with a star(*) are provided in the full version.

3 Ingredients

In this section, we describe three key ingredients used throughout the paper: (1) an anchor pair, (2) a linear-time 10-approximation algorithm, and (3) an ε-certificate.

Throughout this section, we denote by $\Sigma^* = (\sigma_1^*, \sigma_2^*)$ an optimal pair of slabs for the general two-line-center problem, and let $w^* = \max\{w(\sigma_1^*), w(\sigma_2^*)\}$.

3.1 An Anchor Pair and Its Candidate Set

We first define an anchor pair and show that a candidate set for an anchor pair of Σ^* of constant size can be computed in $O(n)$ time. Recall that S denotes the input set of n points in the plane.

▶ **Definition 1** (Anchor pair). *For a pair of slabs $\Sigma = (\sigma_1, \sigma_2)$, we call a pair of points $p, q \in \mathbb{R}^2$ an **anchor pair of Σ** if*
(1) *$p, q \in \sigma_1$ and $d(p, q) \geq \mathrm{diam}((\sigma_1 \setminus \sigma_2) \cap S)/4$, or*
(2) *$p, q \in \sigma_2$ and $d(p, q) \geq \mathrm{diam}((\sigma_2 \setminus \sigma_1) \cap S)/4$.*
Note that p or q may lie in $\sigma_1 \cap \sigma_2$. (See Figure 3.)

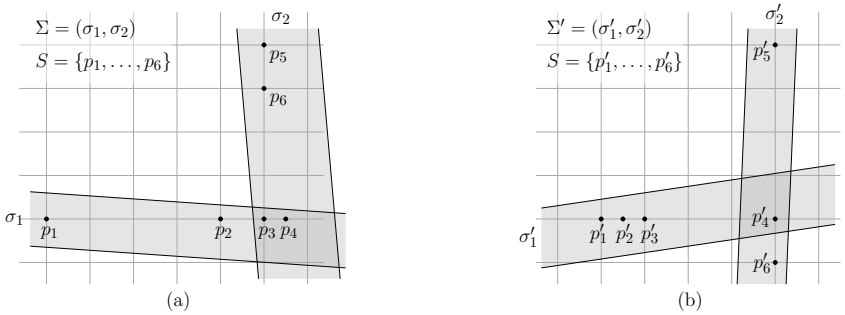


Figure 3 (a) Both (p_2, p_3) and (p_3, p_4) are anchor pairs of Σ, satisfying **(1)** and **(2)**, respectively. (b) (p'_1, p'_2) is an anchor pair of Σ', satisfying **(1)**, but (p'_4, p'_6) is not.

We remark that the authors in [2] used the concept of an anchor pair, defined slightly differently. They showed that a candidate set of constant size can be computed in $O(n \log n)$ time. In the proof of the following lemma, we point out a gap in their proof. By adopting our definition of an anchor pair, we both resolve this gap and obtain its candidate set in $O(n)$ time.

▶ **Lemma 2*.** *We can compute, in $O(n)$ time, a set of at most 11 pairs of points such that for any pair Σ of slabs whose union covers S, at least one of them is an anchor pair of Σ.*

3.2 A Linear-Time 10-Approximation Algorithm

In this section, we give a 10-approximation algorithm for the general two-line-center problem that runs in $O(n)$ time. We remark that the authors in [2] gave a 6-approximation algorithm that runs in $O(n \log n)$ time.

By Lemma 2, we can find a set of at most 11 pairs of points, at least one of which is an anchor pair of Σ^* in $O(n)$ time. For the moment, assume that we have an anchor pair (p, q) of Σ^*. Without loss of generality, assume that $p, q \in \sigma_1^*$ and $d(p, q) \geq \text{diam}((\sigma_1^* \setminus \sigma_2^*) \cap S)/4$.

Let $p_1, p_2, \ldots, p_n$ be the points of S, indexed in increasing order of their distance from ℓ_{pq}. (We do not actually sort them. The indices are used only for reference.) For three points $\alpha, \beta, \gamma \in \mathbb{R}^2$, let $\sigma(\alpha, \beta; \gamma)$ denote the slab whose center line is $\ell_{\alpha\beta}$ and one bounding line passes through γ (see Figure 4).

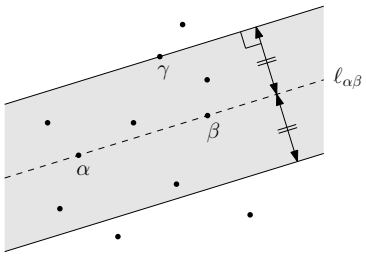


Figure 4 For three points $\alpha, \beta, \gamma \in \mathbb{R}^2$, the gray region represents $\sigma(\alpha, \beta; \gamma)$.

The following lemma implies the existence of a pair of slabs whose maximum width is at most $10w^*$, and one of which is parallel to ℓ_{pq}.

▶ **Lemma 3*.** *There exists a point $r \in S$ such that $\sigma(p, q; r)$ covers all points of $(\sigma_1^* \setminus \sigma_2^*) \cap S$ and $w(\sigma(p, q; r)) \leq 10w^*$. For such r, $\text{width}(S \setminus \sigma(p, q; r)) \leq w^*$.*

Naive Approach and Idea

By Lemma 3, there exists a point $r \in S$ such that both $w(\sigma(p,q;r))$ and $\text{width}(S \setminus \sigma(p,q;r))$ are at most $10w^*$. We now give a brief idea of using a binary search to find such a point r.

Consider the slab $\sigma(p,q;p_i)$ and the set $S \setminus \sigma(p,q;p_i)$ for $1 \le i \le n$. Observe that as i increases, the width $w(\sigma(p,q;p_i))$ increases, while $\text{width}(S \setminus \sigma(p,q;p_i))$ decreases[1]. Using this monotonicity, we can find an index i that minimizes the unimodal function $\max\{w(\sigma(p,q;p_i)), \text{width}(S \setminus \sigma(p,q;p_i))\}$.

More specifically, at each step of the binary search over a range of indices $[s,e] \subseteq [1,n]$, we compute the median index m and evaluate $w(\sigma(p,q;p_m))$ and $\text{width}(S \setminus \sigma(p,q;p_m))$. If $w(\sigma(p,q;p_m)) \ge \text{width}(S \setminus \sigma(p,q;p_m))$, we set the range to $[s,m]$. Otherwise, we set it to $[m,e]$. Let j be the index obtained by the binary search. Observe that j minimizes $\max\{w(\sigma(p,q;p_i)), \text{width}(S \setminus \sigma(p,q;p_i))\}$ over all $1 \le i \le n$. By Lemma 3, we have $\max\{w(\sigma(p,q;p_j)), \text{width}(S \setminus \sigma(p,q;p_j))\} \le 10w^*$.

Note that the time required in each step is dominated by computing $\text{width}(S \setminus \sigma(p,q;p_m))$, which takes $O(n \log n)$ time [22]. Therefore, the overall running time is $O(n \log^2 n)$.

To reduce the time complexity, we avoid computing $width(S \setminus \sigma(p,q;p_m))$ from scratch at every step by sacrificing exactness. We remark that the authors in [2] presented an $O(n \log n)$-time binary search algorithm by reducing the time for each step to $O(n)$. We present an $O(n)$-time binary search algorithm in which each step takes time linear in the size of the current range of indices, even when the points are not sorted. Our algorithm uses the following incremental algorithm that computes a 6-approximation of the width of a point set.

▶ **Theorem 4** (Theorem A.3 of [11]). *Given an online sequence of points in the plane, we can maintain a* 6*-approximation of the width with* $O(1)$ *space and update time.*

We show the following theorem. See the full version for the proof.

▶ **Theorem 5*.** *In* $O(n)$ *time, we can compute a pair of slabs that forms a* 10*-approximation for the general two-line-center problem.*

3.3 An ε-certificate

Before defining an ε-certificate, we first define the notion of an *ε-expansion*. We remark that the definitions in this section are from [3].

▶ **Definition 6** (ε-expansion).

(i) *For an interval* $[a,b]$*, we define its* ε-expansion *to be the interval* $[a - \varepsilon(b-a)/2, b + \varepsilon(b-a)/2]$*. See Figure 5(a).*

(ii) *For a slab centered on a line* ℓ *with width* w*, we define its* ε-expansion *to be the slab centered on the line* ℓ *having width* $(1+\varepsilon)w$*. See Figure 5(b).*

Now we define an ε-certificate in both $\mathbb{R}^1$ and $\mathbb{R}^2$.

▶ **Definition 7** (ε-certificate).

(i) *Let* P *be a set of* n *points on the real line and* $\varepsilon > 0$*. We say that a subset* $Q \subseteq P$ *is an* ε-certificate *of* P *if for any pair of intervals whose union covers* Q*, the union of their* ε*-expansions covers* P*.*

[1] Throughout this section, we refer to a non-decreasing (resp., non-increasing) function simply as an increasing (resp., decreasing) function.

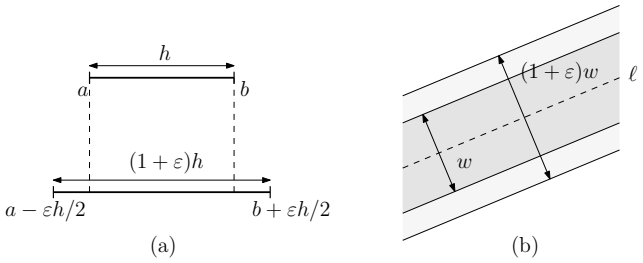


Figure 5 (a) The ε-expansion of an interval. (b) The ε-expansion of a slab.

(ii) *Let P be a set of n points in the plane and $\varepsilon > 0$. We say that a subset $Q \subseteq P$ is an* ε-certificate *of P if for any pair of slabs with the same width whose union covers Q, the union of their ε-expansions covers P. We emphasize that the two slabs must have the same width.*

We note that the existence of an ε-certificate of size $1/\varepsilon^{O(1)}$ was shown in [3]. By proving the following theorem, we refine the hidden constant and show how to construct such an ε-certificate in linear time.

▶ **Theorem 8*.** *For a set S of n points in the plane and $\varepsilon > 0$, an ε-certificate of S of size $O(1/\varepsilon^2)$ can be computed in $O(n)$ time.*

We show that Theorem 8 directly gives a $(1+\varepsilon)$-approximation algorithm that runs in $O(n + \text{poly}(1/\varepsilon))$ time for each of the four problems.

▶ **Theorem 9*.** *For each of the four problems whose optimal solution can be found in $T(n)$ time, we can compute a $(1+\varepsilon)$-approximate solution in $O(n + T(1/\varepsilon^2))$ time.*

For the general two-line-center problem, $T(n) = O(n^2 \log^2 n)$ [19]. By Theorem 9, we obtain a $(1+\varepsilon)$-approximation algorithm running in $O(n+(1/\varepsilon^4)\log^2(1/\varepsilon))$ time. In Section 5, however, we present another algorithm with an improved running time. For the other variants as well, we present $(1+\varepsilon)$-approximation algorithms with better running times than those obtained by applying Theorem 9 with a known exact algorithm as $T(n)$ (see Table 1).

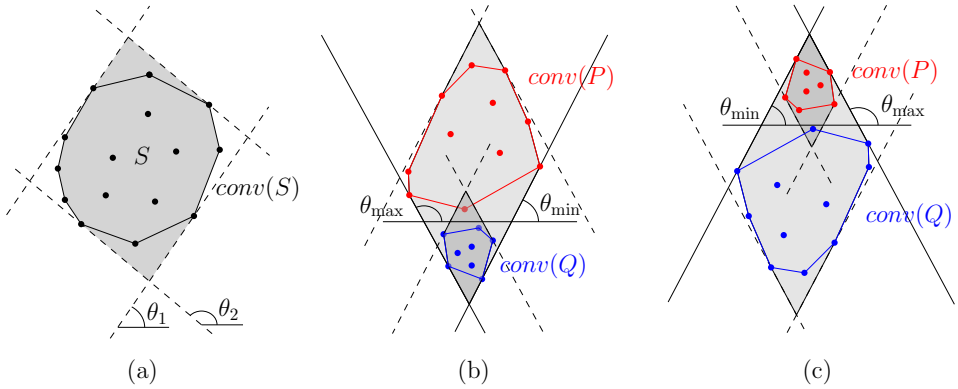
4 One Fixed Orientation

By rotating the coordinate system, we assume without loss of generality that the given orientation θ is 0. Throughout this section, we denote by (σ_1^*, σ_2^*) an optimal pair of slabs for the one-fixed-orientation problem, where σ_1^* is horizontal. Let $w^* = \max\{w(\sigma_1^*), w(\sigma_2^*)\}$.

4.1 Orientation-constrained Width and Dominance

In this section, we recall the concept of an orientation-constrained width and present geometric properties as introduced in [6]. Let $\theta_1 \leq \theta_2$ be two orientations. For a set P of points in the plane, we define the $[\theta_1, \theta_2]$-constrained width of P as $\text{width}_{[\theta_1,\theta_2]}(P) := \min_{\theta\in[\theta_1,\theta_2]} \text{width}_\theta(P)$. We also define $\sigma_{[\theta_1,\theta_2]}(P) := \bigcap_{\theta\in[\theta_1,\theta_2]} \sigma_\theta(P)$ (see Figure 6(a)).

Consider two non-empty sets P and Q of points in the plane that can be separated by a horizontal line, thus $\text{conv}(P) \cap \text{conv}(Q) = \emptyset$. Then, there are exactly two outer common tangents. Let $\theta_{\min} \leq \theta_{\max}$ be the orientations of these lines. We say that P *dominates* Q if $\sigma_{[\theta_{\min},\theta_{\max}]}(Q) \subseteq \sigma_{[\theta_{\min},\theta_{\max}]}(P)$. See Figure 6. By construction, note that either P or Q dominates the other.



(a) (b) (c)

Figure 6 (a) For a point set S and orientations θ_1 and θ_2, the gray region represents $\sigma_{[\theta_1,\theta_2]}(S)$. (b, c) For two point sets P (red) and Q (blue), the two lines represent their outer common tangents. (b) P dominates Q. (c) Q dominates P.

▶ **Lemma 10** ([6]). *With the above notation, suppose that P dominates Q. Then*
(i) $\text{width}_{[\theta_{\min},\theta_{\max}]}(P \cup Q) = \text{width}_{[\theta_{\min},\theta_{\max}]}(P)$, *and*
(ii) *If* $\text{width}(P \cup Q) < d(\text{conv}(P), \text{conv}(Q))$, *then* $\text{width}(P \cup Q) = \text{width}_{[\theta_{\min},\theta_{\max}]}(P)$.

4.2 Decision algorithm

In this section, we present a decision algorithm for a given parameter $\omega > 0$. The algorithm determines whether there exists a pair of slabs of width ω such that one of them is horizontal and their union covers S. If such a pair exists, we report YES. Otherwise, we report NO.

The points $s_1, s_2, \ldots, s_n$ of S are assumed to be sorted in decreasing order of their y-coordinates. For simplicity, we assume that no two input points have the same y-coordinate. Our approach follows that of the previously known algorithm from [6] but removes the dependency on the data structures they used. As a result, we obtain an $O(n)$-time decision algorithm (after sorting), improving upon the previous $O(n \log^2 n)$ bound.

Consider a horizontal slab σ_ω of width ω, and imagine sweeping σ_ω downward over the plane. For a fixed position of σ_ω, let $S_a \subseteq S$ be the set of points strictly above σ_ω, and let $S_b \subseteq S$ be the set of points strictly below σ_ω. Referring to Figure 7, observe that S_b dominates S_a from the beginning until a certain moment, after which S_a dominates S_b until the end, assuming that both S_a and S_b are non-empty. Observe that the answer to the decision problem is YES if and only if there exists a placement of σ_ω such that $\text{width}(S \setminus \sigma_\omega) \leq \omega$ (equivalently, $\text{width}(S_a \cup S_b) \leq \omega$).

Assumption. For the case where the answer is YES, we assume that there exists a placement of σ_ω such that S_a dominates S_b. The other case can be handled symmetrically by sweeping σ_ω upward from the bottom.

Notation. If both S_a and S_b are non-empty, there exist two outer common tangents of them. Observe that these tangent lines do not intersect each other inside σ_ω. Let ℓ_1 denote the right tangent and ℓ_2 denote the left tangent. We denote their orientations by θ_1 and θ_2, respectively. See Figure 7(a). Let $a_1 \in S_a$ and $b_1 \in S_b$ be the points of tangency defining ℓ_1, and similarly, let $a_2 \in S_a$ and $b_2 \in S_b$ be the points of tangency defining ℓ_2. If either S_a or S_b is empty, then ℓ_1 and ℓ_2 (and consequently $\{a_i, b_i, \theta_i \mid i = 1, 2\}$) are undefined.

▶ **Observation 11*.** *With the above notation, S_a dominates S_b if and only if $\theta_1 \leq \theta_2$.*

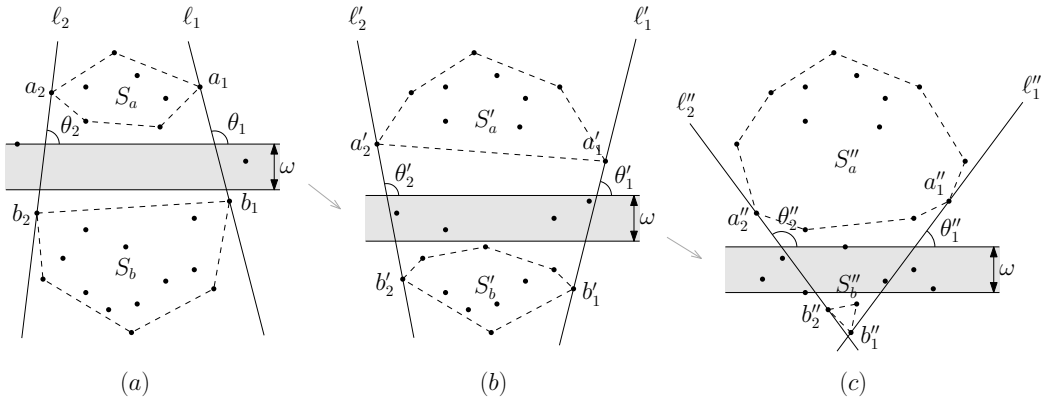


Figure 7 Three snapshots of the sweeping process for a point set. (a) S_a, S_b, and $\{\ell_i, a_i, b_i, \theta_i \mid i = 1, 2\}$ are shown according to the notation defined above. Observe that S_b dominates S_a. (b, c) We use the notations S'_a, S'_b and S''_a, S''_b (and the associated primed and double-primed variables) analogously for the two horizontal slabs positioned below. Observe that S'_a (resp., S''_a) dominates S'_b (resp., S''_b). (a–c) Observe that $\theta_1 \geq \theta'_1 \geq \theta''_1$ and $\theta_2 \leq \theta'_2 \leq \theta''_2$.

4.2.1 Overview of the Algorithm

In this section, we give an overview of the data structures maintained throughout the algorithm and explain how the main loop proceeds. We begin by recalling a data structure for maintaining the convex hull of a point set under a special sequence of insertions and deletions.

▶ **Lemma 12** ([25]). *We can dynamically maintain the convex hull of a set P of points in the plane to support each* window-sliding update *(i.e., either insert a point below all points of P or delete the topmost point of P) in $O(1)$ amortized time. The convex hull can be explicitly reported in $O(h)$ time, where h is the number of vertices of the convex hull.*

We note that in the above data structure, the neighbors of any given vertex on the convex hull can be found in $O(1)$ time. Although it is not explicitly mentioned in [25], it can be achieved since each point p (implicitly) maintains two pointers to its left and right neighbors, respectively, if p is a vertex of the current convex hull (see Sections 3.1 and 4 of [25]).

Observe that as σ_ω moves downward, a point is inserted into S_a below all existing points of S_a, or the topmost point of S_b is deleted from S_b. We refer to each such moment as *an event*. Each event corresponds to a window-sliding update of either S_a or S_b, and we have at most $2n$ events.

Data structures and Main loop

As σ_ω moves downward, we maintain the following using the notation defined above:
- the dynamic data structures for $\text{conv}(S_a)$ and $\text{conv}(S_b)$ (from Lemma 12),
- the tangent lines ℓ_1 and ℓ_2, with their corresponding points a_1, b_1, a_2, and b_2,
- the two antipodal pairs of S_a with respect to the orientations θ_1 and θ_2, respectively.

Initially, we start with a position of σ_ω such that S_a contains only s_1, the topmost point of S. If the corresponding set S_b is empty, we stop the algorithm and report YES. Otherwise, we initialize the data structures for $\text{conv}(S_a)$ and $\text{conv}(S_b)$, respectively. Then we find ℓ_1 and ℓ_2, together with $\{a_i, b_i, \theta_i \mid i = 1, 2\}$. Note that the two antipodal pairs of S_a with respect to θ_1 and θ_2 are both (s_1, s_1).

In the main loop of our decision algorithm, we do the following as σ_ω moves downward. If S_b dominates S_a, we just update the above data structures. Once S_a begins to dominate S_b, we include, for every next event, an additional step that tests whether $\mathrm{width}_{[\theta_1,\theta_2]}(S_a) \leq \omega$, using the maintained data structures. If this condition holds, we stop the algorithm and report YES; otherwise, we proceed the algorithm.

If S_b becomes empty, we set $\theta_1 = 0$ and $\theta_2 = \pi$. We then test whether $\mathrm{width}_{[\theta_1,\theta_2]}(S_a) = \mathrm{width}(S_a) \leq \omega$. If this condition holds, we report YES; otherwise, we report NO.

The correctness of this algorithm, assuming that all data structures are maintained properly and the test $\mathrm{width}_{[\theta_1,\theta_2]}(S_a) \leq \omega$ is evaluated correctly, is guaranteed by the following lemma.

▶ **Lemma 13*.** *For any placement of σ_ω where S_a dominates S_b, we have* $\mathrm{width}(S_a \cup S_b) \leq \omega$ *if and only if* $\mathrm{width}_{[\theta_1,\theta_2]}(S_a) \leq \omega$.

4.2.2 Maintaining Data Structures and Testing the Width Condition

In this section, we describe how to maintain the data structures and test the condition $\mathrm{width}_{[\theta_1,\theta_2]}(S_a) \leq \omega$ for each event that occurs as σ_ω moves downward.

We begin by proving Lemma 14, which motivates the part of the algorithm that tests whether $\mathrm{width}_{[\theta_1,\theta_2]}(S_a) \leq \omega$. Let σ_ω be a horizontal slab of width ω where S_a dominates S_b. Recall that θ_1 and θ_2 are the orientations of the right and left outer tangent lines, respectively. By Observation 11, we have $\theta_1 \leq \theta_2$. Now, let σ'_ω be another horizontal slab of width ω, positioned below σ_ω (the two slabs may partially overlap). We use the notations S'_a, θ'_1, and θ'_2 analogously for σ'_ω. Then, it always holds that $[\theta_1, \theta_2] \subseteq [\theta'_1, \theta'_2]$.

▶ **Lemma 14*.** *With the above notation, assume that* $\mathrm{width}_{[\theta_1,\theta_2]}(S_a) > \omega$. *Then,* $\mathrm{width}_{[\theta'_1,\theta'_2]}(S'_a) \leq \omega$ *if and only if* $\mathrm{width}_{[\theta'_1,\theta_1]}(S'_a) \leq \omega$ *or* $\mathrm{width}_{[\theta_2,\theta'_2]}(S'_a) \leq \omega$.

We first note that the data structures (from Lemma 12) for $\mathrm{conv}(S_a)$ and $\mathrm{conv}(S_b)$ can be maintained in $O(n)$ total time, since each event corresponds to a window-sliding update.

We now describe how to handle each event depending on its type: either an insertion into S_a or a deletion from S_b.

We first observe that if the point being inserted or deleted lies strictly between ℓ_1 and ℓ_2, then nothing changes except the convex hulls (see Figure 8). Therefore, we assume that the point corresponding to an event does not lie strictly between ℓ_1 and ℓ_2.

Insertion into S_a

Let p be the point inserted into S_a, and assume that p lies to the right of ℓ_1 (see Figure 9). The case where p lies to the left of ℓ_2 can be handled symmetrically.

Tangent searching. By our assumption, ℓ_2 remains unchanged, while ℓ_1 changes. Let ℓ'_1 denote the new tangent line, and let θ'_1 denote its orientation. Let $S'_a = S_a \cup \{p\}$. Analogously, let $a'_1 \in S'_a$ and $b'_1 \in S_b$ be the points lying on ℓ'_1. Note that $a'_1 = p$. We find b'_1 by traversing $\mathrm{conv}(S_b)$ clockwise from b_1; we refer to this as the *insertion-type tangent searching procedure*. This procedure takes $O(1+k)$ time, where k is the number of vertices on $\mathrm{conv}(S_b)$ strictly between b_1 and b'_1. We say that these vertices are *involved* in the procedure.

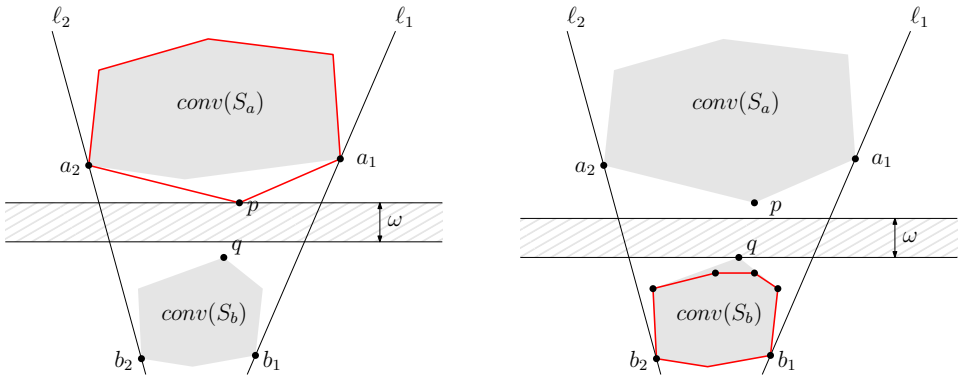


Figure 8 (a) p is inserted into S_a. (b) q is deleted from S_b. The red polygons show the convex hulls immediately after the events. Observe that the tangent lines ℓ_1 and ℓ_2 remain unchanged.

Antipodal pair searching. Now we find the antipodal pairs of S'_a with respect to θ'_1 and θ_2 (since ℓ_2 does not change). Let q be the point such that (a_1, q) is the antipodal pair of S_a with respect to θ_1.

For the orientation θ'_1, observe that $a'_1 (= p)$ is one point of the antipodal pair of S'_a. We can find the other point by traversing $\text{conv}(S'_a)$ clockwise from q. Let q' denote the point such that (a'_1, q') is the antipodal pair of S'_a with respect to θ'_1. We refer to this as the *insertion-type antipodal-pair searching procedure*, which takes $O(1 + k')$ time, where k' is the number of vertices on $\text{conv}(S'_a)$ (equivalently, on $\text{conv}(S_a)$) strictly between q and q'. We say that these vertices are *involved* in the procedure.

For the orientation θ_2, observe that a_2 is always one point of the antipodal pairs for both S_a and S'_a. Let r be the point such that (a_2, r) forms an antipodal pair of S_a with respect to θ_2 (see Figure 9(b)). Let r' be the point such that (a_2, r') forms an antipodal pair of S'_a with respect to θ_2. Then $r' = r$ if p lies to the left of the line of orientation θ_2 passing through r, and $r' = p$ otherwise.

Testing the width condition. If S'_a does not dominate S_b, there is nothing left to do. This can be easily detected by Observation 11. Otherwise, we need to test whether $\text{width}_{[\theta'_1, \theta_2]}(S'_a) \leq \omega$. By Lemma 14, it suffices to determine whether $\text{width}_{[\theta'_1, \theta_1]}(S'_a) \leq \omega$. We do this by following the antipodal pairs of S'_a from the one corresponding to θ'_1 to that of θ_1. Note that p is always one point of the antipodal pairs of S'_a for all orientations between θ'_1 and θ_1, since p lies to the right of ℓ_1. Hence, (p, q) is the antipodal pair of S'_a with respect to θ_1, and (p, q') is the one with respect to θ'_1 as explained above. By traversing $\text{conv}(S'_a)$ counterclockwise starting from q', we can visit all antipodal pairs corresponding to the range of orientations $[\theta'_1, \theta_1]$ in order. Following them sequentially allows us to find an orientation $\theta \in [\theta'_1, \theta_1]$ that minimizes $\text{width}_\theta(S'_a)$. This procedure takes $O(1 + k')$ time, where, as before, k' is the number of vertices on $\text{conv}(S'_a)$ (equivalently, on $\text{conv}(S_a)$) that lie strictly between q and q'. Note that this is bounded by the time required for the insertion-type antipodal-pair searching procedure.

The total time for handling point insertions as σ_ω moves downward depends on the two procedures: the insertion-type tangent searching procedure and the insertion-type antipodal-pair searching procedure. We conclude that it takes $O(n)$ time in total by showing that each input point can be involved at most once in each of the procedures.

▶ **Lemma 15*.** *Each point of S can be involved in the insertion-type tangent searching procedure and the insertion-type antipodal-pair searching procedure at most once, respectively, in the entire algorithm.*

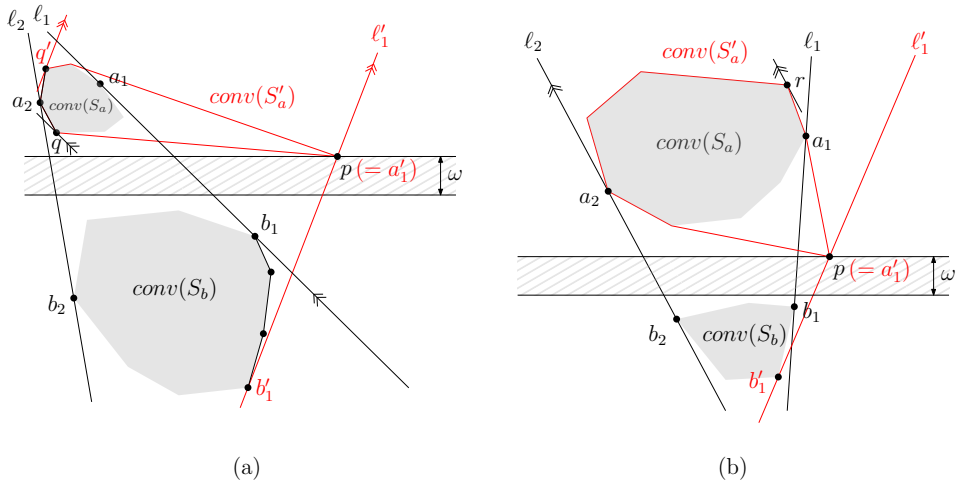


Figure 9 Two cases of insertions. In each figure, the changes after the insertion are shown in red: $\mathrm{conv}(S'_a)$, a'_1, b'_1, and ℓ'_1. In (a), observe that p becomes a point of the antipodal pair of S'_a with respect to θ_2. In contrast, in (b), the antipodal pair of S'_a with respect to θ_2 remains (a_2, r) even after inserting p. In (a), two points are involved in the tangent searching procedure, and one point is involved in the antipodal-pair searching procedure.

In a symmetric way, we handle the deletion of a point from S_b. We provide a brief summary below, and the full details are given in the full version.

Deletion from S_b (Summary)

Let p be the point deleted from S_b. We assume that p lies on ℓ_1, that is $p = b_1$ (see Figure 10).

Tangent searching. By our assumption, ℓ_2 remains unchanged, while ℓ_1 changes. Let ℓ'_1 denote the new tangent line, and let θ'_1 denote its orientation. Define $S'_b = S_b \setminus \{p\}$. Analogously, let $a'_1 \in S_a$ and $b'_1 \in S'_b$ be the points lying on ℓ'_1.

Let t be the counterclockwise neighbor of p on $\mathrm{conv}(S_b)$. We find a'_1 and b'_1 by traversing $\mathrm{conv}(S_a)$ clockwise from a_1 and $\mathrm{conv}(S'_b)$ clockwise from t, simultaneously. We refer to this as the *deletion-type tangent searching procedure*, which takes $O(1 + k_1 + k_2)$ time, where k_1 is the number of points on $\mathrm{conv}(S_a)$ strictly between a_1 and a'_1, and k_2 is the number of points on $\mathrm{conv}(S'_b)$ strictly between t and b'_1. We say that these $k_1 + k_2$ points are *involved* in the procedure.

Antipodal pair searching. We find the antipodal pairs of S_a with respect to θ'_1. (Observe that a'_1 is one point of the antipodal pair.) Let q be the point such that (a_1, q) is the antipodal pair of S_a with respect to θ_1 and let q' be the point such that (a'_1, q') is the antipodal pair of S_a with respect to θ'_1. (see Figure 10). We find q' by traversing $\mathrm{conv}(S_a)$ clockwise from q. We refer to this as the *deletion-type antipodal-pair searching procedure*, which takes $O(1 + k')$ time, where k' is the number of vertices on $\mathrm{conv}(S_a)$ strictly between q and q'. We say that these vertices are *involved* in the procedure.

Testing the width condition. By Lemma 14, it suffices to determine whether $\text{width}_{[\theta_1', \theta_1]}(S_a) \leq \omega$. Recall that (a_1, q) and (a_1', q') are the antipodal pairs of S_a with respect to θ_1 and θ_1', respectively. We do this by traversing $\text{conv}(S_a)$ clockwise, from a_1 to a_1' and from q to q' simultaneously. This takes $O(1 + k_1 + k')$ time, where, as before, k_1 is the number of points on $\text{conv}(S_a)$ strictly between a_1 and a_1', and k' is the number of vertices on $\text{conv}(S_a)$ strictly between q and q'. This is bounded by the total time required by the deletion-type tangent searching procedure and the deletion-type antipodal-pair searching procedure.

The total time for handling point deletions as σ_ω moves downward depends on the two procedures: the deletion-type tangent searching procedure and the deletion-type antipodal-pair searching procedure. We conclude that it takes $O(n)$ time in total by showing that each input point can be involved at most twice in each of the procedures.

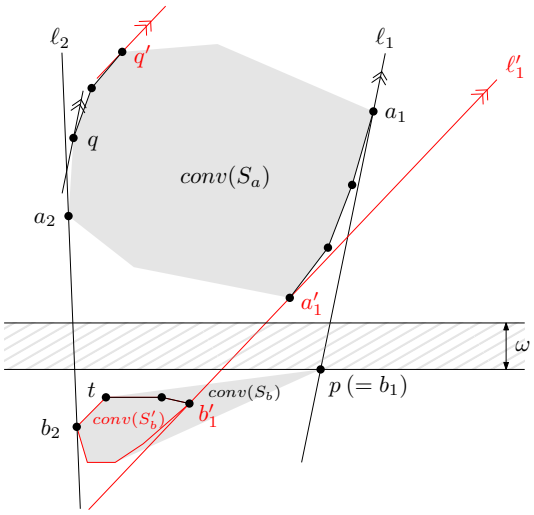


Figure 10 Changes after the deletion of p are shown in red: $\text{conv}(S_b')$, a_1', b_1', and ℓ_1'. Observe that (a_1', q') becomes the antipodal pair of S_a with respect to θ_1' after the deletion. Observe that three points ($k_1 = 2, k_2 = 1$) are involved in the deletion-type tangent searching procedure and one point is involved in the deletion-type antipodal-pair searching procedure.

▶ **Lemma 16*.** *Each point of S can be involved in the deletion-type tangent searching procedure and the deletion-type antipodal-pair searching procedure at most twice, respectively, in the entire algorithm.*

Observe that the initialization of all data structures can be done in $O(n)$ time by Lemma 12. By Lemma 15 and 16, the total time to process all events and to evaluate the condition $\text{width}_{[\theta_1', \theta_1]}(S_a) \leq \omega$ as σ_ω moves downward is $O(n)$. Now we conclude the following theorem.

▶ **Theorem 17.** *Given a set S of n points in the plane, sorted by their y-coordinates, and a real value $\omega > 0$, we can decide in $O(n)$ time whether there exists a pair of slabs of width ω whose union covers S and one of which is horizontal.*

4.3 Optimization Algorithm

We give a brief summary of our approach to obtain Theorem 18. Full details of the algorithm and the proof of its optimality are provided in the full version. We first sort the points of S so that $s_1, \ldots, s_n$ are ordered in decreasing order of their y-coordinates. Let W denote

the set of all pairwise differences between the y-coordinates of points in S. Since W can be represented as a sorted matrix [14], we can identify two consecutive values $w_0, w_1 \in W$ such that $w_0 < w^* \leq w_1$ in $O(n \log n)$ time, by combining our decision algorithm described in Section 4.2 with an efficient selection algorithm for sorted matrices [15].

Observe that the sequence of changes in the sets S_a and S_b that occur as σ_w moves downward is the same for any $w_0 < w < w_1$. Using this observation, we show that w^* can be computed by tracking $width(S_a \cup S_b)$ as σ_w moves downward, for any $w_0 < w < w_1$. We track $width(S_a \cup S_b)$ by following essentially the same procedure as in the decision algorithm. The authors of [6] also use this idea, but their algorithm requires an additional data structure, which results in an extra logarithmic factor.

▶ **Theorem 18*.** *Given a set S of n points in the plane and an orientation θ, we can find a pair of slabs such that one of them has orientation θ and their union covers S while minimizing the maximum of their widths in $O(n \log n)$ time.*

4.4 Approximation Algorithm

Recall that in Section 3.3, we showed that an ε-certificate $Q \subseteq S$ of size $O(1/\varepsilon^2)$ can be computed in $O(n)$ time. After obtaining Q, we find a pair of slabs (σ_1, σ_2) such that their union covers Q, σ_1 has orientation θ, and the maximum of their widths is minimized. Without loss of generality, we assume that σ_1 and σ_2 have the same width (otherwise, we expand the smaller one). Since $Q \subseteq S$, it follows that $w(\sigma_1) = w(\sigma_2) \leq w^*$. Such a pair can be found in $O(|Q| \log |Q|)$ time by Theorem 18.

Let (σ_1', σ_2') be the pair of slabs obtained by taking the ε-expansions of σ_1 and σ_2, respectively. By the definition of an ε-expansion, σ_1' has orientation θ and $w(\sigma_1') = w(\sigma_2') \leq (1+\varepsilon)w^*$. Furthermore, by the definition of an ε-certificate, $\sigma_1' \cup \sigma_2'$ covers S. Therefore, (σ_1', σ_2') forms a $(1+\varepsilon)$-approximate solution.

It takes $O(n + |Q| \log |Q|)$ time to obtain (σ_1', σ_2'). Let $\mu = \min\{n, 1/\varepsilon^2\}$, then we have $|Q| = O(\mu)$. Therefore, the total running time is $O(n + \mu \log \mu)$, which can be expressed as both $O(n \log(1/\varepsilon))$ and $O(n + (1/\varepsilon^2)\log(1/\varepsilon))$. We now conclude the following theorem.

▶ **Theorem 19.** *Given a set S of n points in the plane and an orientation θ, we can find a pair of slabs such that their union covers S, one of them has orientation θ, and the maximum of their widths is at most $(1+\varepsilon)w^*$ in $O(n + \mu \log \mu)$ time, where $\mu = \min\{n, 1/\varepsilon^2\}$.*

5 General Two-Line-Center

Throughout this section, we denote by $\Sigma^* = (\sigma_1^*, \sigma_2^*)$ an optimal pair of slabs for the general two-line-center problem. Let $w^* = \max\{w(\sigma_1^*), w(\sigma_2^*)\}$.

We begin by proving Lemma 20, which will be used to compute a candidate set of orientations for one of the slabs in a $(1+\varepsilon)$-approximate solution.

▶ **Lemma 20*.** *Let p and q be two points in $\mathbb{R}^2$, let $\widetilde{w} > 0$ be a real number, let $c \geq 1$ be a constant, and let $\varepsilon > 0$ be a real number. In $O(1/\varepsilon)$ time, we can compute a set Γ of $O(1/\varepsilon)$ orientations, such that the following holds:*
For every finite set P of points in $\mathbb{R}^2$ and for every slab σ, if

(1) *$P \subset \sigma$,* **(2)** *$p, q \in \sigma$,* **(3)** *$d(p,q) \geq \mathrm{diam}(P)/4$, and* **(4)** *$w(\sigma) \leq \widetilde{w} \leq c \cdot w(\sigma)$,*
then there exists an orientation $\gamma \in \Gamma$ and a slab σ', such that

(1) *$P \subset \sigma'$,* **(2)** *$w(\sigma') \leq (1+\varepsilon) \cdot w(\sigma)$, and* **(3)** *$\sigma'$ has the orientation γ.*

By Lemma 2, we can compute a set F of at most 11 pairs such that one of them is an anchor pair of Σ^* in $O(n)$ time. For the moment, assume that we have an anchor pair (p, q) of Σ^*. (We will repeat the following process for every pair in F.) Without loss of generality, assume that $p, q \in \sigma_1^*$ and $d(p, q) \geq diam((\sigma_1^* \setminus \sigma_2^*) \cap S)/4$.

By Theorem 5, we can compute $\widetilde{w}$ such that $w^* \leq \widetilde{w} \leq 10w^*$ in $O(n)$ time.

Consider a slab σ_0^* that has the same orientation as σ_1^*, has width w^*, and contains σ_1^*. In particular, if $w(\sigma_1^*) = w^*$, then σ_0^* is σ_1^*.

Observe that $P = S \cap (\sigma_1^* \setminus \sigma_2^*)$ and σ_0^* satisfy the four conditions in the statement of Lemma 20 for (p, q) and the value $\widetilde{w}$ computed above, with $c = 10$. Therefore, in $O(1/\varepsilon)$ time, we can compute a set Γ of $O(1/\varepsilon)$ orientations such that there exists a slab σ' with orientation $\gamma \in \Gamma$ that covers $S \cap (\sigma_1^* \setminus \sigma_2^*)$ and satisfies $w(\sigma') \leq (1 + \varepsilon)w^*$. This implies that there exists a pair of slabs whose union covers S, one of which has an orientation from Γ, and whose maximum width is at most $(1 + \varepsilon)w^*$.

For every $\gamma \in \Gamma$, we find a pair of slabs whose union covers S, one of which has orientation γ, and whose maximum width is minimized. By Theorem 18, this can be done in $O((n/\varepsilon) \log n)$ time in total. We repeat this procedure for every pair in F, which adds only a constant factor to the running time. Observe that each of the pairs of slabs we obtain has the property that its union covers S. Among these pairs, we output the one with the smallest maximum width. Thus, we conclude the following lemma.

▶ **Lemma 21.** *We can find a pair of slabs such that their union covers S and the maximum of their widths is at most $(1 + \varepsilon)w^*$ in $O((n/\varepsilon) \log n)$ time.*

To further refine the logarithmic term, we compute an ε-certificate Q of S (see Section 3.3). By Theorem 8, this can be done in $O(n)$ time and $|Q| = O(1/\varepsilon^2)$. We then compute a $(1 + \varepsilon)$-approximate pair (σ_1, σ_2) for the general two-line-center problem on Q, which can be done in $O((|Q|/\varepsilon) \log |Q|)$ time by Lemma 21. Without loss of generality, we assume that σ_1 and σ_2 have the same width (otherwise, we expand the smaller one). Let $w = w(\sigma_1) = w(\sigma_2)$. Since $Q \subseteq S$, it is clear that $w \leq (1 + \varepsilon)w^*$.

Let (σ_1', σ_2') be the pair of slabs obtained by taking the ε-expansions of σ_1 and σ_2, respectively. By definition, $\sigma_1' \cup \sigma_2'$ covers S, and $w(\sigma_1') = w(\sigma_2') \leq (1 + \varepsilon)^2 w^* \leq (1 + 3\varepsilon)w^*$.

The total running time is $O(n + (\mu/\varepsilon) \log \mu)$, where $\mu = \min\{n, 1/\varepsilon^2\}$, which can also be expressed as both $O((n/\varepsilon) \log(1/\varepsilon))$ and $O(n + (1/\varepsilon^3) \log(1/\varepsilon))$. By replacing ε with $\varepsilon/3$, we obtain the following theorem.

▶ **Theorem 22.** *Given a set S of n points in the plane, we can find a pair of slabs such that their union covers S and the maximum of their widths is at most $(1+\varepsilon)w^*$ in $O(n+(\mu/\varepsilon) \log \mu)$ time, where $\mu = \min\{n, 1/\varepsilon^2\}$.*

References

1. Pankaj K. Agarwal and Cecilia Magdalena Procopiuc. Approximation algorithms for projective clustering. *J. Algorithms*, 46(2):115–139, 2003. doi:10.1016/S0196-6774(02)00295-X.
2. Pankaj K. Agarwal, Cecilia Magdalena Procopiuc, and Kasturi R. Varadarajan. A $(1+\varepsilon)$-approximation algorithm for 2-line-center. *Comput. Geom.*, 26(2):119–128, 2003. doi:10.1016/S0925-7721(03)00017-8.
3. Pankaj K. Agarwal, Cecilia Magdalena Procopiuc, and Kasturi R. Varadarajan. Approximation algorithms for a k-line center. *Algorithmica*, 42(3-4):221–230, 2005. doi:10.1007/S00453-005-1166-X.
4. Charu C. Aggarwal, Cecilia Magdalena Procopiuc, Joel L. Wolf, Philip S. Yu, and Jong Soo Park. Fast algorithms for projected clustering. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 61–72. ACM Press, 1999. doi:10.1145/304182.304188.

5 Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 70–81. ACM, 2000. doi:10.1145/342009.335383.

6 Taehoon Ahn and Sang Won Bae. Constrained two-line center problems. In *35th International Symposium on Algorithms and Computation, ISAAC 2024, Sydney, Australia, December 8-11, 2024*, volume 322 of *LIPIcs*, pages 5:1–5:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ISAAC.2024.5.

7 Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 250–257. ACM, 2002. doi:10.1145/509907.509947.

8 Sang Won Bae. Minimum-width double-strip and parallelogram annulus. *Theor. Comput. Sci.*, 833:133–146, 2020. doi:10.1016/J.TCS.2020.05.045.

9 Gill Barequet and Sariel Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38(1):91–109, 2001. doi:10.1006/JAGM.2000.1127.

10 Kaushik Chakrabarti and Sharad Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 89–100. Morgan Kaufmann, 2000. URL: http://www.vldb.org/conf/2000/P089.pdf.

11 Timothy M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom.*, 35(1-2):20–35, 2006. doi:10.1016/J.COMGEO.2005.10.002.

12 Chaeyoon Chung, Taehoon Ahn, Sang Won Bae, and Hee-Kap Ahn. Parallel line centers with guaranteed separation. *Comput. Geom.*, 129:102185, 2025. doi:10.1016/J.COMGEO.2025.102185.

13 Vasek Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979. doi:10.1287/MOOR.4.3.233.

14 Greg N. Frederickson and Donald B. Johnson. The complexity of selection and ranking in X+Y and matrices with sorted columns. *J. Comput. Syst. Sci.*, 24(2):197–208, 1982. doi:10.1016/0022-0000(82)90048-4.

15 Greg N. Frederickson and Donald B. Johnson. Generalized selection and ranking: Sorted matrices. *SIAM J. Comput.*, 13(1):14–30, 1984. doi:10.1137/0213002.

16 Sariel Har-Peled and Kasturi R. Varadarajan. Projective clustering in high dimensions using core-sets. In *Proceedings of the 18th Annual Symposium on Computational Geometry, Barcelona, Spain, June 5-7, 2002*, pages 312–318. ACM, 2002. doi:10.1145/513400.513440.

17 Refael Hassin and Nimrod Megiddo. Approximation algorithms for hitting objects with straight lines. *Discret. Appl. Math.*, 30(1):29–42, 1991. doi:10.1016/0166-218X(91)90011-K.

18 Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh S. Vempala. Locality-preserving hashing in multidimensional spaces. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 618–625. ACM, 1997. doi:10.1145/258533.258656.

19 Jerzy W. Jaromczyk and Miroslaw Kowaluk. The two-line center problem from a polar view: A new algorithm and data structure. In *Algorithms and Data Structures, 4th International Workshop, WADS '95, Kingston, Ontario, Canada, August 16-18, 1995, Proceedings*, volume 955 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1995. doi:10.1007/3-540-60220-8_47.

20 D. T. Lee and Ying-Fung Wu. Geometric complexity of some loction problems. *Algorithmica*, 1(2):193–211, 1986. doi:10.1007/BF01840442.

21 Nimrod Megiddo and Arie Tamir. On the complexity of locating linear facilities in the plane. *Oper. Res. Lett.*, 1(5):194–197, 1982. doi:10.1016/0167-6377(82)90039-6.

22 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction*. Texts and Monographs in Computer Science. Springer, New Haven, CT, USA, 1985. UMI Order No. 78-18546. doi:10.1007/978-1-4612-1098-6.

23 Cecilia M. Procopiuc. Projective clustering. In *Encyclopedia of Machine Learning and Data Mining*, pages 1018–1025. Springer, 2017. doi:10.1007/978-1-4899-7687-1_676.

24 Cecilia Magdalena Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002*, pages 418–427. ACM, 2002. doi:10.1145/564691.564739.

25 Haitao Wang. Dynamic convex hulls under window-sliding updates. *Comput. Geom. Topol.*, 4(1):3:1–3:14, 2025. doi:10.57717/CGT.V4I1.53.

Tensor Computation of Euler Characteristic Functions and Transforms

Jessi Cisewski-Kehe
Department of Statistics, University of Wisconsin, Madison, WI, USA

Brittany Terese Fasy
Montana State University, Bozeman, MT, USA

Alexander McCleary
School of Computing, Montana State University, Bozeman, MT, USA

Eli Quist
School of Computing, Montana State University, Bozeman, MT, USA

Abstract

The weighted Euler characteristic transform (WECT) and Euler characteristic function (ECF) have proven to be useful tools in a variety of applications. However, current methods for computing these functions are either not optimized for GPU computation or do not scale to higher-dimensional settings. In this work, we present a tensor-based framework for computing such topological descriptors which is highly optimized for GPU architectures and works in full generality across simplicial and cubical complexes of arbitrary dimension. Experimentally, the framework demonstrates significant speedups over existing methods when computing the WECT and ECF across a variety of two- and three-dimensional datasets. Computation of these transforms is implemented in a publicly available Python package called `pyECT`.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases Topological data analysis, weighted Euler characteristic transform, Euler characteristic function, tensor computation, GPU computation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.32

Supplementary Material *Software (Source code)*: `https://doi.org/10.5281/zenodo.18877833`

Funding This project began as part of a collaborative NSF FRG grant (grant numbers 1854336, 1854220, and 2038556).
Jessi Cisewski-Kehe: NSF grants 2337243 and 2346683.
Brittany Terese Fasy: NSF grants 2046730 and 2152737.
Alexander McCleary: NSF grants 2046730 and 2152737.
Eli Quist: NSF grants 2046730 and 2152737.

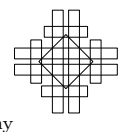
Acknowledgements We thank Jack Ruder and Jacob Sriraman for their valuable assistance in developing and refining portions of the methodology and code used in this work. Computational efforts were performed on the Tempest High Performance Computing System, operated and supported by the University Information Technology Research Cyberinfrastructure (RRID:SCR_026229) at Montana State University.

1 Introduction

The field of topological data analysis (TDA) utilizes invariants from topology to study the shape of data. Among the invariants employed are homology and the original topological invariant, the Euler characteristic [10]. In practice, a single invariant is not typically robust enough to capture the shape of data. For this reason, filtrations (i.e., nested sequences of



© Jessi Cisewski-Kehe, Brittany Terese Fasy, Alexander McCleary, and Eli Quist;
licensed under Creative Commons License CC-BY 4.0



shapes) are often constructed from data and topological invariants are then applied. This allows for significantly more information to be retained, leading to constructions such as persistence diagrams [9] and Euler characteristic functions [21].

Topological transforms [24] further extend the idea of applying topological invariants to filtrations by constructing a parameterized family of filtrations and then applying topological invariants. The starting point for topological transforms is typically a geometric simplicial or cubical complex K in $\mathbb{R}^n$. For each direction vector $\mathbf{s} \in \mathbb{S}^{n-1}$, a directional filtration of K is constructed and then a topological invariant is applied to that filtration. Applying the Euler characteristic (EC) to a single directional filtration results in an Euler characteristic function (ECF), while applying it to each directional filtration in $\mathbb{S}^{n-1}$ results in the Euler characteristic transform (ECT) [15, 24]. The ECT is a *complete invariant*: a geometric simplicial complex K in $\mathbb{R}^n$ can be reconstructed solely from its ECT [11, 19]. In other words, no information is lost when passing from the simplicial complex K to its ECT.

Useful variants of the ECF and ECT are the weighted Euler characteristic function (WECF) and the weighted Euler characteristic transform (WECT) for weighted complexes [12]. Much like the ECT, the WECT is a complete invariant for weighted geometric complexes [12]. Moreover, the WECT has been employed in a broad range of applications, including: in oncology for predicting survival rates of patients with glioblastoma multiforme brain tumors [6], in plant biology for classifying seed phenotypes [1], and in image classification tasks [3, 4, 12]. Numerous software packages already exist for computing the WECT and its variants, including those presented in [1, 6, 13, 14, 17, 18]. While many of these packages provide useful tools, they are either not optimized for GPU architectures or are limited in the types (or dimension) of complexes they can process.

In this work, we provide a fully tensor-based framework for computing WECFs and WECTs. Our implementation applies to arbitrary geometric simplicial and cubical complexes in Euclidean spaces. The tensor operations we use are highly optimized for fast computation on modern GPU hardware, providing significant speedups over existing implementations.

2 Background

In this section, we provide necessary foundations in computational topology and tensor computation to develop the main results. For simplicity, we restrict our attention to (abstract) simplicial complexes; however, our results and code also work for cubical complexes.

For a simplicial complex K and for any $i \in \{0, 1, \ldots, \dim(K)\}$, we denote the set of i-simplices of K by K_i. We use the notation $\mathrm{sub}(K)$ for the set of subcomplexes of K. Furthermore, we define $k = |K|$ and $k_i = |K_i|$.

▶ **Definition 1** (Lower-Star Filtrations). *Let K be a simplicial complex with vertex set K_0. A* vertex filter *on K is a function $f\colon K_0 \to \mathbb{R}$. For a vertex filter $f\colon K_0 \to \mathbb{R}$, the* lower-star filtration *induced by f is the function $F\colon \mathbb{R} \to \mathrm{sub}(K)$, defined as*

$$F(t) := \{\sigma \in K \mid \max_{v \in \sigma} f(v) \leq t\}. \tag{1}$$

The main examples of lower-star filtrations used in this paper are directional filtrations. These are defined for any (abstract) simplicial complex K with a function $\ell\colon K_0 \to \mathbb{R}^n$. In practice, simplicial complexes are often naturally embedded in $\mathbb{R}^n$ and, in this setting, the function ℓ simply maps each vertex to its coordinates.

▶ **Definition 2** (Vertex Height Functions and Directional Filtrations). *Let K be a simplicial complex with a map $\ell\colon K_0 \to \mathbb{R}^n$ and let $\mathbf{s} \in \mathbb{S}^{n-1}$ be a direction vector. The* vertex height function *in direction $\mathbf{s}$ is the map $h_{\mathbf{s}}\colon K_0 \to \mathbb{R}$ defined as $h_{\mathbf{s}}(v) := \ell(v) \cdot \mathbf{s}$. The* directional filtration *of K in direction $\mathbf{s}$, denoted as $H_{\mathbf{s}}$, is the lower-star filtration of K induced by $h_{\mathbf{s}}$.*

2.1 Weighted Euler Characteristic Functions and Transforms

Here, we introduce weighted Euler characteristic functions and transforms. For a more thorough review, see the survey paper [15].

▶ **Definition 3** (Weighted Simplicial Complexes). *A* weighted simplicial complex *is a pair* (K, ω)*, where* K *is a simplicial complex and* $\omega \colon K \to \mathbb{R}$ *is a function called a* weight function*.*

Weighted simplicial complexes are often used to model various types of data. For example, triangular meshes are widely used to model three-dimensional surfaces, with weight functions capturing scalar fields on the surface. Another example comes from image data, where grayscale images are converted into weighted simplicial complexes via the *Freudenthal triangulation* [4, 12] with weights derived from the pixel intensities.

▶ **Definition 4** (Weighted Euler Characteristic). *The* weighted Euler characteristic (WEC) *of a weighted simplicial complex* (K, ω) *is*

$$\chi(K, \omega) := \sum_{\sigma \in K} (-1)^{\dim \sigma} \omega(\sigma).$$

Note that the Euler characteristic is a special case of the WEC, where the weight function is equal to the constant function $\mathbf{1} \colon K \to \mathbb{R}$. While the WEC is a remarkably useful invariant given its simplicity, there is limited information that a single number can capture about a simplicial complex. To incorporate more information, we apply the WEC to a filtration:

▶ **Definition 5** (Weighted Euler Characteristic Function). *Let* $\mathbb{K} = (K, \omega)$ *be a weighted simplicial complex and let* $f \colon K_0 \to \mathbb{R}$ *be a vertex filter of* K *with lower-star filtration* F*. The* weighted Euler characteristic function[1] *of* $\mathbb{K}$ *induced by* f *is the function* $\mathrm{WECF}_f \colon \mathbb{R} \to \mathbb{R}$ *defined as*

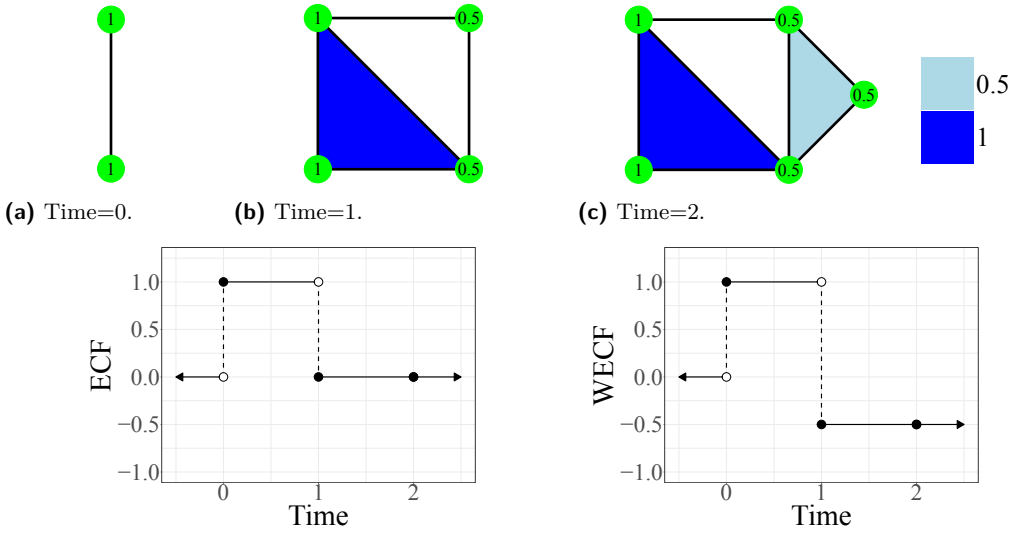
$$\mathrm{WECF}_f(t) := \chi\big(F(t), \omega|_{F(t)}\big). \tag{2}$$

▶ **Definition 6** (Directional WECF). *Let* $\mathbb{K} = (K, \omega)$ *be a weighted simplicial complex with a function* $\ell \colon K_0 \to \mathbb{R}^n$ *and let* $\mathbf{s} \in \mathbb{S}^{n-1}$ *be a direction vector. The* directional WECF *of* $\mathbb{K}$ *in direction* $\mathbf{s}$ *is the WECF of* $\mathbb{K}$ *induced by the height function* $h_{\mathbf{s}}$*.*

▶ Remark 7. In image analysis tasks (and their three-dimensional analogs), there are two common (W)ECFs considered. The first (implemented in [13, 25, 26]) uses the pixel intensities as the vertex filter. We refer to the resulting ECF simply as the ECF of an image. The second, which we refer to as the directional WECF of an image, uses the pixel intensities as weights and the height function $h_{\mathbf{s}}$ for a given direction vector $\mathbf{s} \in \mathbb{S}^1$ as the vertex filter. This method is implemented in [1, 14, 23].

▶ **Example 8.** An illustration of the directional ECF and directional WECF is presented in Figure 1. The full simplicial complex, including weighted vertices, is displayed in Figure 1(c). The filtration is along direction $(1, 0)$, and the corresponding ECF and WECF are included in Figure 1(d) and Figure 1(e), respectively. The ECF and WECF are the same until time 1 when two new vertices are introduced with weights of 0.5.

[1] Weighted Euler characteristic functions are also known as weighted Euler characteristic curves. Due to their general lack of continuity, we use the term function rather than curve.



(a) Time=0. **(b)** Time=1. **(c)** Time=2.

(d) ECF in direction (1,0). **(e)** WECF in direction (1,0).

Figure 1 Filtration of a simplicial complex (a)–(c) along direction (1,0), with vertex weights shown in green discs. The weight assigned to a simplex is the maximum weight of its vertices. The corresponding ECF and WECF are displayed in (d) and (e) respectively.

Typically, there is no canonical direction vector to choose when computing directional WECFs. Hence, it is common to choose *all* directions, resulting in the weighted Euler characteristic transform (WECT).

▶ **Definition 9** (Weighted Euler Characteristic Transform). *Let* $\mathbb{K} = (K, \omega)$ *be a weighted simplicial complex with a map* $\ell\colon K_0 \to \mathbb{R}^n$. *The weighted Euler characteristic transform of* $\mathbb{K}$ *is the function* $\mathrm{WECT}_{\mathbb{K}}\colon \mathbb{S}^{n-1} \times \mathbb{R} \to \mathbb{R}$ *defined as*

$$\mathrm{WECT}_{\mathbb{K}}(\mathbf{s}, t) := \mathrm{WECF}_{h_{\mathbf{s}}}(t) = \chi\big(H_{\mathbf{s}}(t), \omega\big). \tag{3}$$

2.2 Existing Frameworks for Computation of the ECF and (W)ECT

Due to its simplicity and ease of computation, the WECT and its variants are widely used in applied settings (resulting in many available software packages). `Demeter` [1] is a Python package that computes the directional ECF and ECT of axis-aligned cubical complexes. Note that it incorporates an optimization of bucket sorting the filtration values of each vertex. However, this package does not work for generic simplicial complexes, and only computes the special case of the unweighted ECT. Next, the `SINATRA` package [23] is an R package that computes the directional ECF and ECT through an algorithm similar to `Demeter`. However, unlike `Demeter`, `SINATRA` works for arbitrary, unweighted simplicial complexes. Note that as `Demeter` and `SINATRA` are not optimized for speed, they are not included in the experimental comparison. Third, `EuCalc`[2] [14] is a highly optimized package that uses a preprocessing step to compute critical points of the directional ECF. This allows for efficient computation of the exact directional ECF (as opposed to a vectorization of it). The package only accepts axis-aligned cubical complexes as input, but does generalize

[2] `https://github.com/HugoPasse/Eucalc`

to weighted complexes. Finally, the `DECT` package[3] [17] is a Python package designed to compute the differentiable ECT (DECT) of simplicial complexes. While the DECT is a slightly different variant, the package also includes an implementation for computing the ECT of simplicial or cubical complexes. Although it does not allow for weighted complexes, this package is designed for GPU architectures, making it an interesting point of comparison. Furthermore, [18] provides further optimizations for computing the DECT on CUDA GPUs, but is consequently a far less general framework.

The `FastTopology` framework[4] [13] is a highly optimized implementation for computing the ECF of lower-star filtrations of two- and three-dimensional axis-aligned cubical complexes. While it is designed to be highly parallelizable on CPU architectures, it does not directly support vectorized computation (e.g., on a GPU) or the WECF. `GPU ECC` [25, 26] is a GPU-based implementation of the ECF. Unlike the other existing packages, it is designed for GPU architectures. However, it is highly optimized for image data, so it only works for axis-aligned cubical complexes and does not generalize to weighted complexes. Additionally, it does not compute the directional ECF or ECT, and currently is only available as an implementation on Windows systems (which is unavailable in our computation environment).

The framework presented in this paper, `pyECT`, admits several advantages over these existing packages for computing the WECT and ECF. First, unlike all existing frameworks (except `SINATRA`), `pyECT` works in full generality across arbitrary weighted simplicial and cubical complexes; see Table 1. Second, `pyECT` provides methods for computing both the WECFs of lower-star filtrations and WECTs of weighted geometric complexes. This generality is not available in any of these existing packages. Third, the tensor operations used in `pyECT` are highly optimized for modern GPU hardware, resulting in significant speedups over these existing packages. The `GPU ECC` package admits this advantage as well, but is far less general as it only accepts axis-aligned cubical complexes and does not compute the directional WECF or WECT. Finally, the tensor-based approach used in `pyECT` yields simple code (our core WECT algorithm is around 25 lines of code) that is easily readable, maintable, and customizable.

One minor disadvantage of `pyECT` is that it does not compute exact EC values at all heights of the directional ECF (which is available in `EuCalc`). While computing exact values retains more information and can be an advantage for reconstruction tasks, it leads to difficulties in vectorizing the (W)ECT. However, vectorizations of the WECT are in fact more desirable for many real-world applications, especially when used as input to distance-based methods or machine learning pipelines (such as those in [1, 12, 15, 24]).

Table 1 Comparison of supported input types for ECT packages.

Package	Simplicial Complexes	Cubical Complexes	Axis-Aligned Cubical Complexes
`pyECT`	✓ yes	✓ yes	✓ yes
`EuCalc`	✗ no	✗ no	✓ yes
`FastTopology`	✗ no	✗ no	✓ yes
`Demeter`	✗ no	✗ no	✓ yes
`SINATRA`	✓ yes	✓ yes	✓ yes

[3] `https://github.com/aidos-lab/dect`
[4] `https://github.com/zavalab/ML/tree/master/FastTopology`

2.3 Tensor Operations

Our algorithm almost exclusively uses tensor operations. These operations have highly optimized implementations in PyTorch [2, 16] that run in parallel on both CPUs and GPUs. Here, we review tensors and the tensor operations used in our algorithm.

For any positive integer m, let $[[m]] := \{0, 1, \dots, m-1\}$. For $\{m_i\}_{i=0}^{d-1} \subset \mathbb{Z}_+$, a d-dimensional tensor of shape $(m_0, m_1, \dots, m_{d-1})$ is a function

$$T\colon [[m_0]] \times [[m_1]] \times \cdots \times [[m_{d-1}]] \to \mathbb{R}.$$

We exclusively use one-dimensional tensors (vectors), two-dimensional tensors (matrices), and three-dimensional tensors. A specific dimension of a tensor is called an *axis*. We now introduce the tensor operations utilized in our algorithm.

1. Zeros: Given a shape L, $\texttt{zeros}(L)$ is the tensor with shape L, where every value is 0.
2. Element-wise operations: Any function $f\colon \mathbb{R} \to \mathbb{R}$ can be applied to tensors element-wise by post-composition. That is, if T is a tensor, then $f(T)$ is the tensor with the same shape as T and values $f(T)[i_0, i_1, \dots, i_{d-1}] = f\big(T[i_0, i_1, \dots, i_{d-1}]\big)$.
3. Matrix multiplication: If T and S are tensors with shape (m_0, m_1) and (m_1, m_2) respectively, we use the notation $T * S$ to denote their matrix multiplication.
4. Cumulative sum: Let T be a tensor of shape (m_0, m_1). Then, $\texttt{CumulativeSum}(T)$ is the tensor with shape (m_0, m_1) and values

$$\texttt{CumulativeSum}(T)[i, j] = \sum_{\substack{k \in [[m_1]] \\ k \le j}} T[i, k].$$

   In this work, we use the cumulative sum on two-dimensional tensors along the second axis. However, the cumulative sum can be defined over any axis of a tensor.
5. Reduce maximum: Let T be a tensor of shape $(m_0, m_1, \dots, m_{d-1})$ and let $p \in [[d]]$. Then, $\texttt{ReduceMax}(T, p)$ is the tensor with shape $(m_0, \dots, m_{p-1}, m_{p+1}, \dots, m_{d-1})$ and values

$$\texttt{ReduceMax}(T, p)[i_0, \dots, i_{p-1}, i_{p+1}, \dots i_{d-1}] = \max_{j \in [[m_p]]} T[i_0, \dots, i_{p-1}, j, i_{p+1}, \dots i_{d-1}].$$

6. Advanced indexing: Let T be a tensor with shape (m_2, m_3), and let I be a tensor with shape (m_0, m_1) and values in $[[m_2]]$. The tensor $T[I]$ has shape (m_0, m_1, m_3) and values

$$T[I][i, j, k] = T\big[I[i, j], k\big].$$

7. Scatter add: Let S be a tensor with shape (m_0, m_1), let I be a tensor with shape (m_0, m_2) and values in $[[m_1]]$, and let T be a tensor with shape (m_2). The function $\texttt{ScatterAdd}(I, T)$ updates S in-place by adding to it the difference tensor D with shape (m_0, m_1) and values

$$D[i, j] = \sum_{\substack{k \in [[m_2]] \\ I[i,k] = j}} T[k]. \tag{4}$$

   That is, for a tensor S, we have $\texttt{ScatterAdd}(I, T)(S) = S + D$, where $+$ is taken element-wise and S is updated in place. In practice, the tensor D is not explicitly built.

3 Computation of WECFs

In this section, we present a new algorithm for computing the WECFs of a weighted simplicial complex with respect to a finite set of vertex filters. Computation of the WECT of a weighted geometric simplicial complex is a special case of Algorithm 1, and corresponds to the case where the vertex filters are height functions; see Example 10.

We fix a weighted simplicial complex $\mathbb{K} = (K, \omega)$ and a set of vertex filters $\{f_0, f_1, \ldots, f_{m-1}\}$ for the entirety of this section. To simplify notation, we extend each $f_p \colon K_0 \to \mathbb{R}$ to K by defining $f_p^{\max} \colon K \to \mathbb{R}$ as

$$f_p^{\max}(\sigma) = \max_{v \in \sigma} f_p(v).$$

We fix enumerations $K_0 = \{v_0, v_1, \ldots, v_{k_0-1}\}$ and $K_i = \{\sigma_0^i, \sigma_1^i, \ldots, \sigma_{k_i-1}^i\}$ for each $i \geq 1$.

Our algorithm uses the following inputs

- `Complex`: A list of tuples:
  - `Complex`$[0] =$ (`Filters`, `VertexWeights`)
    - `Filters` is a tensor with shape (k_0, m) and values `Filters`$[a, p] = f_p(v_a)$.
    - `VertexWeights` is a tensor with shape (k_0) and values `VertexWeights`$[a] = \omega(v_a)$.
  - for $i \geq 1$: `Complex`$[i] =$ (i-`SimplexVertices`, i-`SimplexWeights`)
    - i-`SimplexVertices` is a tensor with shape $(k_i, i+1)$ and values in $[[k_0]]$. The rows of this tensor correspond to the vertex sets of each i-simplex.
    - i-`SimplexWeights` is a tensor with shape (k_i) and values i-`SimplexWeights`$[b] = \omega(\sigma_b^i)$.
- `NumVals`: The number of values to sample the filter functions over.

In order to discretize the filtration height parameter, define `MaxVal` as

$$\texttt{MaxVal} = \max_{\substack{p \in [[m]] \\ v \in K_0}} |f_p(v)|.$$

We discretize $[-\texttt{MaxVal}, \texttt{MaxVal}]$ by evenly sampling `NumVals` points. That is, we define the map $\beta \colon [[\texttt{NumVals}]] \to [-\texttt{MaxVal}, \texttt{MaxVal}]$ by

$$\beta(q) = q \frac{2 * \texttt{MaxVal}}{\texttt{NumVals} - 1} - \texttt{MaxVal}.$$

The function β has a left inverse $\alpha \colon [-\texttt{MaxVal}, \texttt{MaxVal}] \to [[\texttt{NumVals}]]$ defined as

$$\alpha(t) = \left\lceil \frac{(\texttt{NumVals} - 1)(\texttt{MaxVal} + t)}{2 * \texttt{MaxVal}} \right\rceil. \tag{5}$$

The maps α and β are order-preserving and satisfy

$$\alpha(t) \leq q \iff t \leq \beta(q) \tag{6}$$

for all $t \in [-\texttt{MaxVal}, \texttt{MaxVal}]$ and $q \in [[\texttt{NumVals}]]$.[5]

Here, we prove that Algorithm 1 produces the tensor `WECFs` with shape $(m, \texttt{NumVals})$ and values

$$\texttt{WECFs}[p, q] = \mathrm{WECF}_{f_p}\big(\beta(q)\big). \tag{7}$$

[5] In other words, α and β form a Galois connection.

Algorithm 1 `ComputeWECFs(Complex, NumVals)`.

Input: `Complex`, `NumVals`
Output: `WECFs`: a tensor of shape $(m, \texttt{NumVals})$ and values $\texttt{WECFs}[p,q] = \mathrm{WECF}_{f_p}(\beta(q))$.

```
Filters, VertexWeights ← Complex[0]
diffWECFs ← zeros(m, NumVals)

VIndices ← α(Filters)
ScatterAdd(VIndices^T, VertexWeights)(diffWECFs)

for i = 1 to dim(K) do
    SimplexVertices, SimplexWeights ← Complex[i]
    SimpIndices ← VIndices[SimplexVertices]
    MaxSimpIndices ← ReduceMax(SimpIndices, 1)

    ScatterAdd(MaxSimpIndices^T, (−1)^i * SimplexWeights)(diffWECFs)
end for

WECFs ← CumulativeSum(diffWECFs)
return WECFs
```

The algorithm begins by initializing a zero-tensor `diffWECFs` with shape $(m, \texttt{NumVals})$. Next, the vertex filter values are mapped to indices in $[[\texttt{NumVals}]]$ by applying the map α elementwise to the tensor `Filters`. This gives the tensor `VIndices` with values

$$\texttt{VIndices}[a,p] = \alpha\big(f_p(v_a)\big). \tag{8}$$

The operation $\texttt{ScatterAdd}(\texttt{VIndices}^T, \texttt{VertexWeights})$ is then applied to `diffWECFs`, resulting in adding to `diffWECFs` the tensor D_0 with shape $(m, \texttt{NumVals})$ and values

$$D_0[p,q] = \sum_{\substack{a \in [[k_0]] \\ \texttt{VIndices}[a,p]=q}} \texttt{VertexWeights}[a].$$

Next, inside the for-loop, fix an $i \in \{1, 2, \ldots, \dim(K)\}$. The tensor `SimplexVertices` has rows indexed by K_i with the entries in each row corresponding to the vertex sets of simplices in K_i. Therefore, the tensor `MaxSimpIndices` has shape (k_i, m) and values

$$\begin{aligned}
\texttt{MaxSimpIndices}[b,p] &= \max_{j \in [[i+1]]} \texttt{VIndices}\big[\texttt{SimplexVertices}[b,j], p\big] \\
&= \max_{v \in \sigma_b^i} \alpha\big(f_p(v)\big) \\
&= \alpha\big(\max_{v \in \sigma_b^i} f_p(v)\big) \qquad \text{as } \alpha \text{ is order-preserving} \\
&= \alpha\big(f_p^{\max}(\sigma_b^i)\big).
\end{aligned} \tag{9}$$

The operation $\texttt{ScatterAdd}(\texttt{MaxSimpIndices}^T, (-1)^i * \texttt{SimplexWeights})$ updates `diffWECFs` by adding to it the tensor D_i with shape $(m, \texttt{NumVals})$ and values

$$D_i[p,q] = \sum_{\substack{b \in [[k_i]] \\ \texttt{MaxSimpIndices}[b,p]=q}} (-1)^i \texttt{SimplexWeights}[b].$$

After the conclusion of the for loop, we have $\texttt{diffWECFs} = \sum_{i=0}^{\dim(K)} D_i$. Finally, a cumulative sum is applied to `diffWECFs` to obtain the tensor `WECFs`. From the fact that the cumulative sum is a linear operator, we have

$$\texttt{WECFs} = \sum_{i=0}^{\dim(K)} \texttt{CumulativeSum}(D_i).$$

The cumulative sum of D_0 has values

$$\begin{aligned}
\texttt{CumulativeSum}(D_0)[p,q] &= \sum_{q' \le q} \sum_{\substack{a \in [[k_0]] \\ \texttt{VIndices}[a,p] = q'}} \texttt{VertexWeights}[a] \\
&= \sum_{\substack{a \in [[k_0]] \\ \texttt{VIndices}[a,p] \le q}} \texttt{VertexWeights}[a] \\
&= \sum_{\substack{a \in [[k_0]] \\ \alpha(f_p(v_a)) \le q}} \texttt{VertexWeights}[a] && \text{by Equation (8)} \\
&= \sum_{\substack{a \in [[k_0]] \\ f_p(v_a) \le \beta(q)}} \texttt{VertexWeights}[a] && \text{by Equation (6)} \\
&= \sum_{\substack{v \in K_0 \\ f_p(v) \le \beta(q)}} \omega(v). && (10)
\end{aligned}$$

Likewise, for any $i \in \{1, \dots, \dim(K)\}$, Equation (9) together with the same argument used to show Equation (10) gives

$$\texttt{CumulativeSum}(D_i)[p,q] = \sum_{\substack{\sigma \in K_i \\ f_p^{\max}(\sigma) \le \beta(q)}} (-1)^i \omega(\sigma).$$

Therefore, the tensor `WECFs` has values

$$\begin{aligned}
\texttt{WECFs}[p,q] &= \sum_{\substack{v \in K_0 \\ f_p(v) \le \beta(q)}} \omega(v) + \sum_{i=1}^{\dim(K)} \sum_{\substack{\sigma \in K_i \\ f_p^{\max}(\sigma) \le \beta(q)}} (-1)^i \omega(\sigma) \\
&= \sum_{\substack{\sigma \in K \\ f_p^{\max}(\sigma) \le \beta(q)}} (-1)^{\dim(\sigma)} \omega(\sigma) \\
&= \mathrm{WECF}_{f_p}\left(\beta(q)\right).
\end{aligned}$$

▶ **Example 10.** Computing a discretization of the WECT of a weighted simplicial complex $\mathbb{K} = (K, \omega)$ with a map $\ell \colon K_0 \to \mathbb{R}^n$ is a special case of Algorithm 1 as follows. Consider a tensor D, with shape (d, n), of direction vectors over which to sample the WECT. For any $p \in [[d]]$, let $\mathbf{s}_p \in \mathbb{R}^n$ be the p-th row of D. Let V be the tensor, with shape (k_0, n), of the vertex coordinates of K. Then, the tensor $V * D^T$ has values

$$(V * D^T)[a,p] = h_{\mathbf{s}_p}(v_a).$$

Therefore, using the tensor $V * D^T$ as the `Filters` tensor in Algorithm 1 produces a discretization of the WECT over the finite sample of directions in D.

3.1 Complexity Analysis

Here, we analyze the asymptotic complexity of Algorithm 1. Assume that the ambient dimension n is fixed. Let k be the total number of simplices of K, let m be the number of vertex filters, and let `NumVals` be the number of values to sample the vertex filters over.

The input `Complex` requires $\Theta(km)$ memory usage. The algorithm first initializes a zero tensor `diffWECFs` of shape $(m, \texttt{NumVals})$, requiring $\Theta(m\texttt{NumVals})$ memory. The function α (see Equation (5)) is then applied elementwise to `Filters`, taking $\Theta(km)$ time.

Next, the operation $\texttt{ScatterAdd}(\texttt{VIndices}^T, \texttt{VertexWeights})$ is applied to `diffWECFs`. ScatterAdd adds a value from `VertexWeights` to `diffWECFs` for each entry of $\texttt{VIndices}^T$, resulting in a time complexity of $\Theta(km)$ and no additional memory usage as the operation is performed in place.

Now, for a fixed dimension $i \in \{1, 2, \dots n\}$, the tensor

$$\texttt{SimpIndices} = \texttt{VIndices}[\texttt{SimplexVertices}]$$

with shape $(k_i, i+1, m)$ is constructed, taking $\Theta(km)$ time and memory. This tensor is then reduced along the second axis by taking the maximum values, creating the tensor `MaxSimpIndices` with shape (k_i, m) in $\Theta(km)$ time and memory. The tensor `diffWECFs` is then updated by applying the map

$$\texttt{ScatterAdd}(\texttt{MaxSimpIndices}^T, (-1)^i * \texttt{SimplexWeights})$$

which takes $\Theta(km)$ time and memory.

Thus, we have a total time and memory complexity of $\Theta(nkm) = \Theta(km)$ after conclusion of the for-loop. Finally, a cumulative sum is applied to `diffWECFs`, taking $\Theta(m\texttt{NumVals})$ time and using no additional memory. This results in $\Theta(m(k + \texttt{NumVals}))$ for Algorithm 1 as the final time and memory complexity. In practice, m and `NumVals` are typically held constant. Indeed, this is necessary to achieve consistent vectorizations of data. Holding m and `NumVals` constant results in a final time and memory complexity of $\Theta(k)$ for Algorithm 1.

4 Experiments

In order to demonstrate the efficacy of our implementation, we evaluate and compare the runtime of Algorithm 1 with existing methods (see Section 2.2) on a variety of datasets.[6]

4.1 Experimental Design

Experiments were performed on a single node with a single CPU or a single GPU (unless otherwise specified). EPYC Rome CPUs and NVIDIA A100 GPUs were used. Runtimes for CPU experiments were recorded using the `time` Python library using the performance counter for benchmarking. For the GPU experiments, runtimes were recorded using `PyTorch`'s `torch.cuda.Event` API to prevent inaccurate timing from asynchronous GPU operations. The methods were evaluated on the following two- and three-dimensional datasets:

- **Fashion-MNIST** [27]. Fashion-MNIST is a dataset consisting of 28×28 grayscale images of clothing items. Its training partition consists of 60000 images with equal representation

[6] The implementation of `pyECT` can be accessed here: `https://anonymous.4open.science/r/pyECT-anonymous-902F/`. The source code for reproducing experiments (with instructions) is available here: `https://anonymous.4open.science/r/pyECT-experiments-anonymous-0E4F/`.

across ten classes (6000 images per class). In our experiments, we use a random subset of 1000 images sampled uniformly without replacement from the training partition, which are then converted to cubical complexes.

- **ImageNet** [7, 20]. The ImageNet Sample Image dataset [20], which consists of 1000 samples from the Large Scale Visual Recognition Challenge (ILSVRC) 2012-2017 subset of the ImageNet dataset [7] was used. All images in the sample were converted to grayscale using the ITU-R 601-2 luma transform within the Pillow Python library [5]. Then, the images were padded to the maximum dimensions of all images in the dataset, resulting in each image having uniform size of 1632×2400 pixels. The padded pixels were assigned a pixel intensity value of 1.0 (on a zero to one scale), and the images were converted to cubical complexes.
- **3DCCs**. A synthetic dataset of 1000 three-dimensional cubical complexes was generated. Each complex has dimension $100 \times 100 \times 100$, with pixel intensities drawn i.i.d. from the uniform distribution on $[0, 1]$.
- **Stanford Bunny and Armadillo** [22]. The Stanford 3D Scanning Repository provides triangulated mesh models of three-dimensional objects. We use the Bunny and Armadillo models. The meshes were converted to (unweighted) simplicial complexes after preprocessing to de-duplicate vertices, edges, and faces. In the resulting simplicial complexes, the Bunny dataset consists of 2503 vertices, 7473 edges, and 4968 faces. The Armadillo is significantly larger, consisting of 172974 vertices, 518916 edges, and 345944 faces.

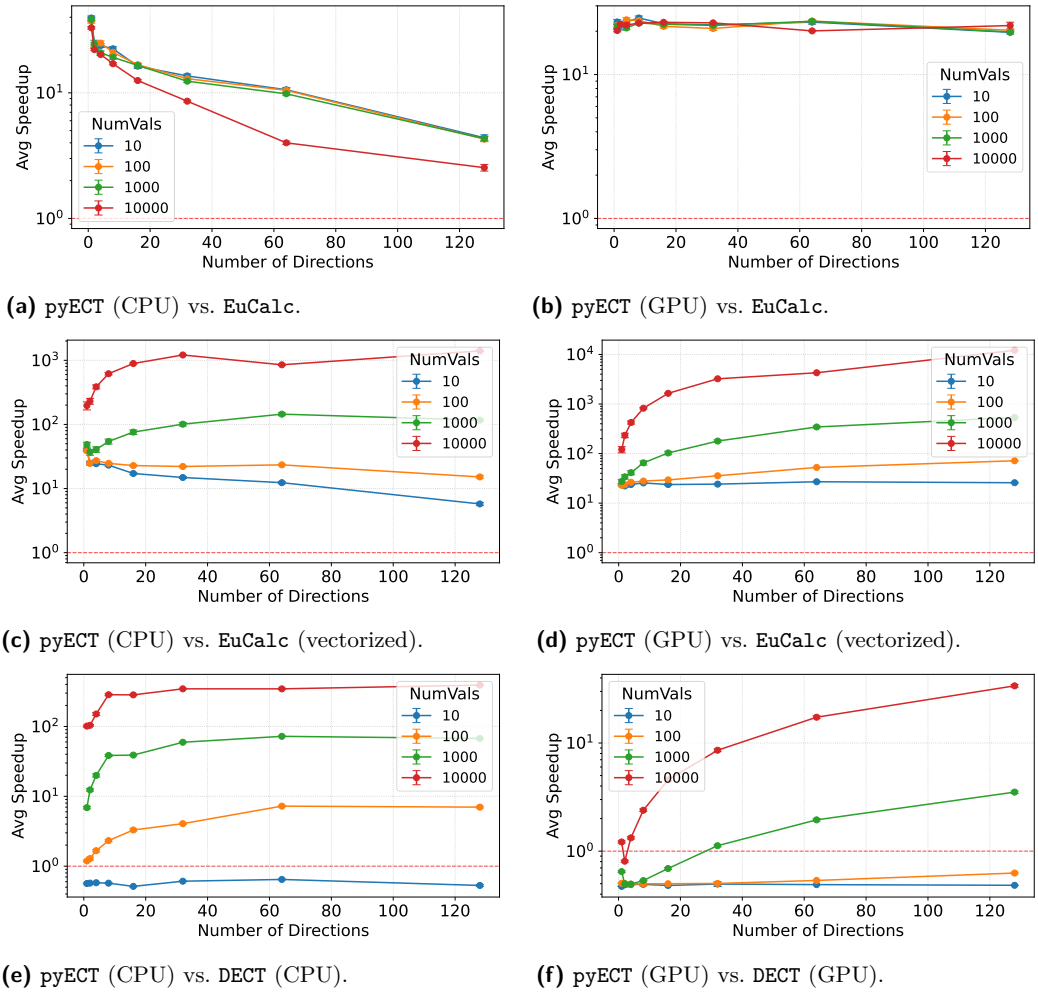
4.2 Results

We now present the experimental results, measured by the average speedup factor of our method over existing methods. The average speedup factor is defined as the runtime of the baseline method (e.g., `EuCalc`, `DECT`, or `FastTopology`) divided by the runtime of the proposed method, `pyECT`, averaged across all images in the dataset. Across all figures, error bars represent 95% confidence intervals.

WECT Results

In this section, we present results on the average speedup of `pyECT` over `EuCalc` and `DECT` when computing the WECT of the Fashion-MNIST and ImageNet datasets. As `EuCalc` does not provide a GPU implementation, all results for `EuCalc` are measured on the CPU. Also, as `EuCalc` does not natively provide a vectorized result, we provide results both including and excluding the vectorization time.

The results for Fashion-MNIST are shown in Figure 2. Across all settings, `pyECT` demonstrates speedups over both `EuCalc` and `DECT`. When comparing against `EuCalc` (Figure 2(a)–(d)), `pyECT` demonstrates a speedup in both CPU and GPU implementations – even without the vectorization step. However, the GPU implementation of `pyECT` significantly outperforms the CPU implementation as the number of directions increases, which is especially pronounced when including the vectorization time for `EuCalc`. As expected, when increasing the number of values used in the filtration, the speedup factor of `pyECT` over `EuCalc` increases drastically. When comparing against `DECT` (subfigures (e) and (f)), `pyECT` also demonstrates significant speedups. While the computation speed is comparable for low numbers of directions and filtration heights, `pyECT` significantly outperforms `DECT` for larger choices of parameters. Also, as `DECT` is optimized for GPU architectures, the average speedup on GPU architectures is less pronounced than that of other comparisons. However, when constrained to CPU architectures, `pyECT` is much faster, indicating another advantage of `pyECT`'s general applicability for all scenarios.



(a) `pyECT` (CPU) vs. `EuCalc`.

(b) `pyECT` (GPU) vs. `EuCalc`.

(c) `pyECT` (CPU) vs. `EuCalc` (vectorized).

(d) `pyECT` (GPU) vs. `EuCalc` (vectorized).

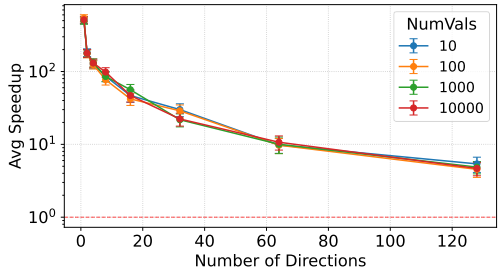
(e) `pyECT` (CPU) vs. `DECT` (CPU).

(f) `pyECT` (GPU) vs. `DECT` (GPU).

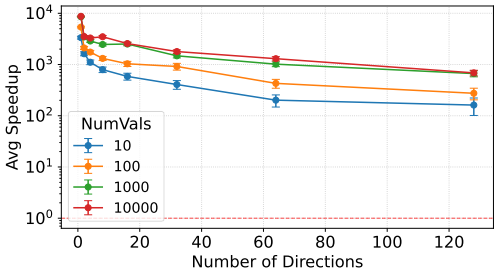
Figure 2 A comparison for computing on the Fashion-MNIST dataset for WECT computation. Figures (a) and (b) compare `pyECT` with `EuCalc` excluding the vectorization time, (c) and (d) compare `pyECT` with `EuCalc` including the vectorization time, and (e) and (f) compare `pyECT` with `DECT`. CPU implementations of `pyECT` are shown on the left and GPU implementations are shown on the right. Vertical axes are not consistent across subfigures. In nearly all cases, `pyECT` outperforms the baseline methods. This behavior is especially pronounced when using the GPU architecture with large numbers of directions and filtration heights.

The average speedup results on the ImageNet are shown in Figure 3. When comparing against `EuCalc` (both with and without the vectorization step, in Figures 3a–3d), `pyECT` demonstrates a significant speedup in both CPU and GPU implementations. This speedup is especially pronounced for the GPU implementation of `pyECT`, as it is much more stable to large numbers of directions and filtration height values. Due to the large size of the complexes in this dataset, the CPU speedup of `pyECT` is less pronounced than on the Fashion-MNIST dataset (especially for large numbers of directions and heights). Furthermore, the vectorization time of `EuCalc` is less impactful on this dataset, as the vectorization step is less significant compared to the overall runtime for the large complexes. In the GPU implementation of `pyECT`, we observe a trend where the average speedup increases as the number of height filtration values increases. While this behavior appears counterintuitive, in reality the vectorized code encounters less efficient memory access patterns at lower numbers

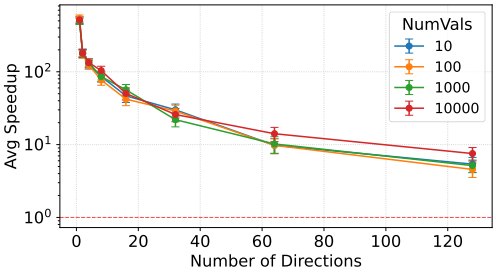
of heights, which leads to a smaller advantage over `EuCalc` with the smaller tensor values. For a technical description of memory access patterns specific to the GPU hardware used in these experiments, we refer the reader to [8]. In the comparison against `DECT` (Figures 3e and 3f), `pyECT` again demonstrates significant speedups. In fact, due to the large size of the complexes in this dataset, `DECT` was unable to compute the WECT for large numbers of directions and height values (due to memory constraints on the GPU). This results in the appearance of missing values in Figures 3e and 3f. Note, however, that `pyECT` is still able to compute the WECT in these scenarios, demonstrating its memory efficiency. Furthermore, in the cases where the WECT could be computed by `DECT`, the `pyECT` implementation outperforms `DECT` for larger numbers of directions and height filtration values.



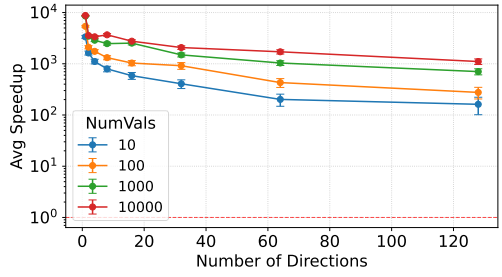
(a) `pyECT` (CPU) vs. `EuCalc`.



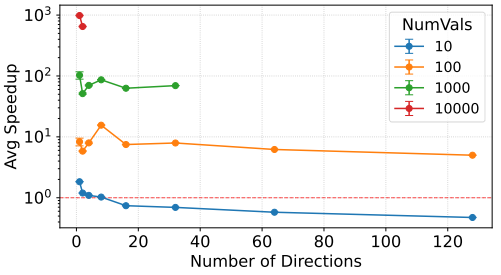
(b) `pyECT` (GPU) vs. `EuCalc`.



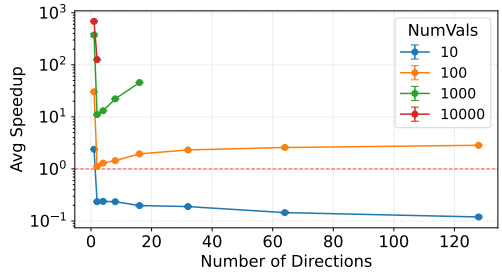
(c) `pyECT` (CPU) vs. `EuCalc` (vectorized).



(d) `pyECT` (GPU) vs. `EuCalc` (vectorized).



(e) `pyECT` (CPU) vs. `DECT` (CPU).

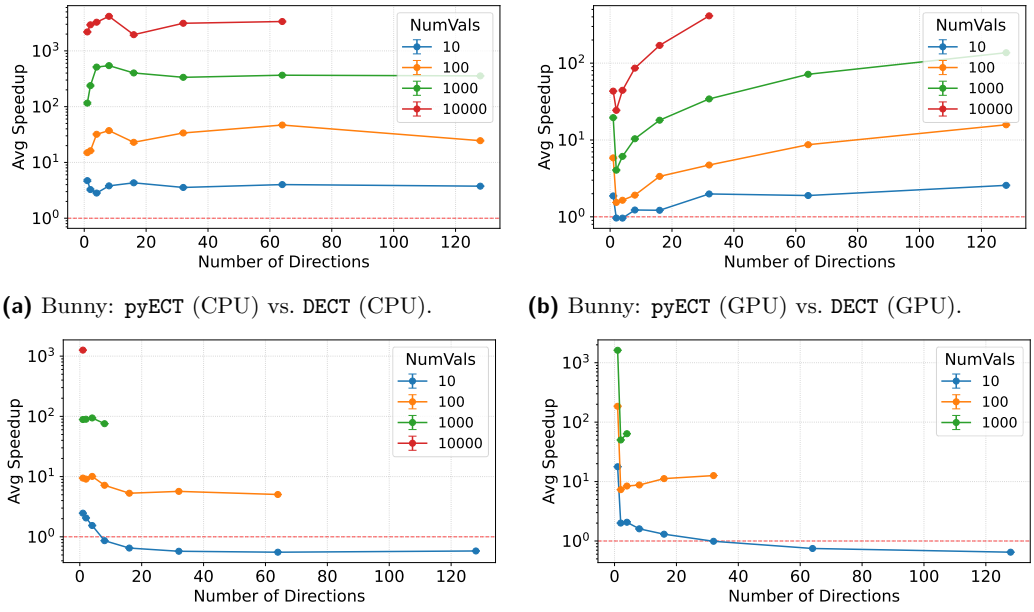


(f) `pyECT` (GPU) vs. `DECT` (GPU)

Figure 3 A comparison for computing on the ImageNet dataset for WECT computation. Figures (a) and (b) compare `pyECT` with `EuCalc` excluding the vectorization time, (c) and (d) compare `pyECT` with `EuCalc` including the vectorization time, and (e) and (f) compare `pyECT` with `DECT`. Note that vertical axes are not consistent across subfigures. `DECT` was unable to compute the WECT for large numbers of directions and heights due to memory constraints, resulting in missing values in (e) and (f). However, `pyECT` was able to compute the WECT in these scenarios.

Speedup results for the comparison of `pyECT` against `DECT` on the three-dimensional mesh datasets are presented in Figure 4. Note that experiments were run 10 times on each mesh dataset to generate the average and confidence interval values shown in the plots.

For large choices of directions and filtration heights in both datasets, `DECT` was unable to compute the ECT due to memory constraints. While this results in missing values in all subplots, from the cases where `DECT` was able to compute the ECT, `pyECT` outperforms `DECT` (especially for larger choices of parameters). The memory issues are more pronounced in the Armadillo dataset (due to its larger size). Note that while `pyECT` demonstrates significant speedups in the GPU implementations, its advantage is even more pronounced in the CPU implementations, highlighting the versatility of `pyECT` across different hardware architectures.



(a) Bunny: `pyECT` (CPU) vs. `DECT` (CPU).

(b) Bunny: `pyECT` (GPU) vs. `DECT` (GPU).

(c) Armadillo: `pyECT` (CPU) vs. `DECT` (CPU).

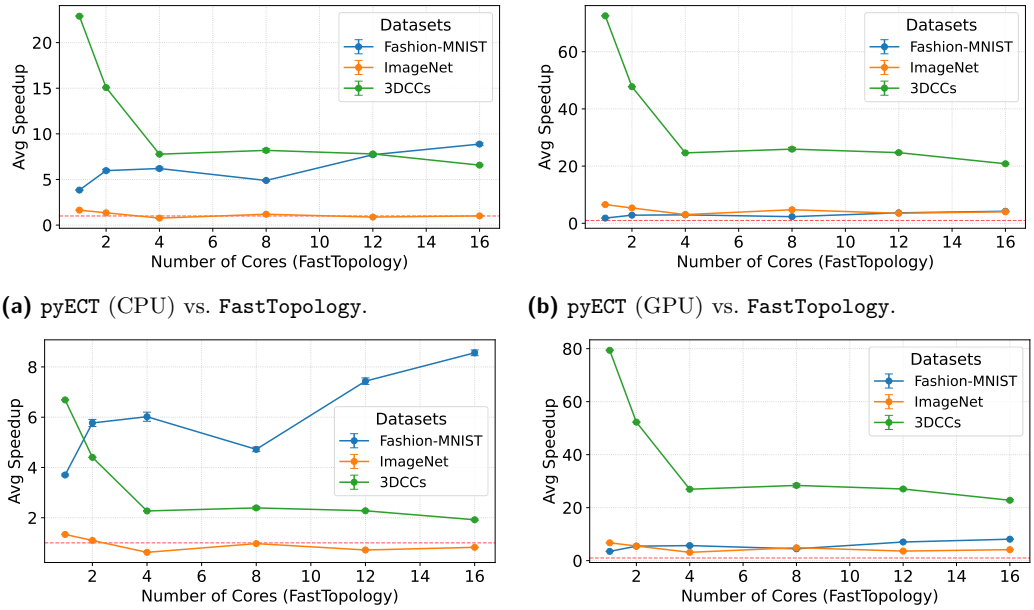
(d) Armadillo: `pyECT` (GPU) vs. `DECT` (GPU).

Figure 4 A comparison of `pyECT` against `DECT` for computing the ECT on the Stanford Bunny and Armadillo meshes. Note that vertical axes are not consistent across subfigures. In many cases, `DECT` was unable to compute the ECT due to memory constraints (leading to missing values in the plots). However, `pyECT` was able to compute the ECT in all scenarios, and especially outperforms `DECT` for large numbers of directions and filtration heights.

ECF Results

In this section, we present the speedup of `pyECT` over `FastTopology` when computing the ECF of the Fashion-MNIST, ImageNet, and 3DCCs datasets. Due to the consistent computation graph structure of the ECF algorithm, `pyECT` is able leverage the PyTorch TorchDynamo compilation framework. These compiled runtimes are also measured and compared with `FastTopology`. As `FastTopology` is parallelizable across multiple CPU cores, we include results for `FastTopology` using 1, 2, 4, 8, 12, and 16 CPU cores. Note that `FastTopology` does not provide a GPU implementation. The results are presented in Figure 5. We observe that `pyECT` provides a significant speedup over `FastTopology` in nearly all cases. When constrained to CPU architectures, `pyECT` has a less pronounced speedup, but is still comparable in speed on a single CPU core when compared with the fully parallelized `FastTopology` implementation. When utilizing GPU architectures, `pyECT` demonstrates a significant speedup over the fully parallelized `FastTopology`, especially on the three-dimensional 3DCCs dataset. The compilation strategy further increases the speed of `pyECT` on GPU architectures, but adds unrewarded overhead on CPU architectures.

Next, to verify the efficiency of the computation on the GPU, we compute the ECF of 2D and 3D images with intensity drawn uniformly and measure throughput as in [25, 26]. For 2D images (sizes 2048^2, 4096^2, and 8192^2) with data pre-loaded into GPU memory, uncompiled `pyECT` achieves 1.11–1.13 GPix/s at 1,000 thresholds and 2.40–2.52 GPix/s at 10,000 thresholds, while the compiled version achieves 1.28–1.32 GPix/s and 2.52–4.11 GPix/s, respectively. For 3D images (sizes 128^3, 256^3, and 512^3), uncompiled `pyECT` achieves 0.48–0.49 GVox/s at 1,000 thresholds and 1.05–1.09 GVox/s at 10,000 thresholds, while the compiled version achieves 0.43–0.56 GVox/s and 1.09–1.76 GVox/s, respectively.



(a) `pyECT` (CPU) vs. `FastTopology`.

(b) `pyECT` (GPU) vs. `FastTopology`.

(c) `pyECT` (compiled, CPU) vs. `FastTopology`.

(d) `pyECT` (compiled, GPU) vs. `FastTopology`.

Figure 5 A comparison of average speedup of `pyECT` over `FastTopology` when computing the ECF of multiple datasets. The comparison evaluates both uncompiled and compiled versions of `pyECT`. Note that vertical axes are not consistent across subfigures.

5 Discussion

In this work, we present a tensor-based framework for computing WECFs (including ECFs and the WECT). This framework allows for efficient computation of these functions using tensor operations, which are well-suited for modern GPU hardware. Furthermore, our method works in full generality for weighted simplicial complexes (or cubical complexes) of arbitrary dimension. Our framework is implemented in a publicly available Python package called `pyECT`. Experimentally, `pyECT` demonstrates a significant speedup over existing methods across a variety of datasets.

The `pyECT` package also contains a variety of tools for constructing and working with simplicial/cubical complexes. Our package includes fully vectorized methods for building weighted Freudenthal complexes and weighted cubical complexes from images. Additionally, `pyECT` has dedicated methods for computing ECFs of images and WECTs of weighted geometric simplicial complexes in $\mathbb{R}^n$. These two special cases are implemented as `PyTorch` neural network modules. Future work includes using this framework in differentiable forms

of topological transforms (as in [17]) and incorporating these transforms into neural network architectures. Additionally, we believe that this work will encourage further investigation into vectorized methods for computing other topological functions and transforms.

References

1 Erik J. Amézquita, Michelle Y. Quigley, Tim Ophelders, Jacob B. Landis, Daniel Koenig, Elizabeth Munch, and Daniel H. Chitwood. Measuring hidden phenotype: quantifying the shape of barley seeds using the Euler characteristic transform. *in silico Plants*, 4(1):diab033, 2022.

2 Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 929–947, 2024.

3 Leo M. Betthauser. *Topological Reconstruction of Grayscale Images*. Ph.d. dissertation, University of Florida, 2018. doi:10.5555/AAI29374331.

4 Jessi Cisewski-Kehe, Brittany Terese Fasy, Dhanush Giriyan, and Eli Quist. The weighted Euler characteristic transform for image shape classification. arXiv preprint, 2023. arXiv:2307.13940.

5 Jeffrey A. Clark and Contributors. Pillow: the friendly pil fork. https://python-pillow.github.io/, 2024. Version 10.3.0.

6 Lorin Crawford, Anthea Monod, Andrew X Chen, Sayan Mukherjee, and Raúl Rabadán. Predicting clinical outcomes in glioblastoma: an application of topological and functional data analysis. *Journal of the American Statistical Association*, 115(531):1139–1150, 2020.

7 Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi:10.1109/CVPR.2009.5206848.

8 Rajeshwari Devaramani, Jonathan Bentz, and Tony Scudiero. Unlock gpu performance: Global memory access in cuda, 2025. Accessed on October 31, 2025. URL: https://developer.nvidia.com/blog/unlock-gpu-performance-global-memory-access-in-cuda/.

9 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002. doi:10.1007/S00454-002-2885-2.

10 Leonhard Euler. Elementa doctrinae solidorum. *Novi Commentarii Academiae Scientiarum Petropolitanae*, pages 109–140, 1758.

11 Robert Ghrist, Rachel Levanger, and Huy Mai. Persistent homology and Euler integral transforms. *Journal of Applied and Computational Topology*, 2(1):55–60, 2018. doi:10.1007/s41468-018-0017-1.

12 Qitong Jiang, Sebastian Kurtek, and Tom Needham. The weighted Euler curve transform for shape and image analysis. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3685–3694, Los Alamitos, CA, USA, June 2020. IEEE Computer Society. doi:10.1109/CVPRW50498.2020.00430.

13 Daniel J. Laky and Victor M. Zavala. A fast and scalable computational topology framework for the Euler characteristic. *Digital Discovery*, 3:392–409, 2024.

14 Vadim Lebovici, Steve Oudot, and Hugo Passe. Efficient Computation of Topological Integral Transforms. In Leo Liberti, editor, *22nd International Symposium on Experimental Algorithms (SEA 2024)*, volume 301 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SEA.2024.22.

15 Elizabeth Munch. An invitation to the Euler characteristic transform. *The American Mathematical Monthly*, 132(1):15–25, 2025. doi:10.1080/00029890.2024.2409616.

16 Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

17 Ernst Röell and Bastian Rieck. Differentiable Euler characteristic transforms for shape classification. In *The Twelfth International Conference on Learning Representations*, 2024. URL: https://openreview.net/forum?id=MO632iPq3I.

18 Udit Saxena. Scalable gpu-accelerated Euler characteristic curves: Optimization and differentiable learning for pytorch. arXiv preprint, 2025. arXiv:2510.20271.

19 P. Schapira. Tomography of constructible functions. In Gérard Cohen, Marc Giusti, and Teo Mora, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 427–435, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. doi:10.1007/3-540-60114-7_33.

20 Eli Schwartz. imagenet-sample-images: 1000 images, one per imagenet class. https://github.com/EliSchwartz/imagenet-sample-images, 2020. GitHub repository, accessed 24 Nov 2025.

21 Jean Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.

22 Stanford University Computer Graphics Laboratory. Stanford 3d scanning repository. https://graphics.stanford.edu/data/3Dscanrep/. Accessed: May 21, 2026.

23 Wai Shing Tang, Gabriel Monteiro da Silva, Henry Kirveslahti, Erin Skeens, Bibo Feng, Timothy Sudijono, Kevin K. Yang, Sayan Mukherjee, Brenda Rubenstein, and Lorin Crawford. A topological data analytic approach for discovering biophysical signatures in protein dynamics. *PLOS Computational Biology*, 18(5):1–42, May 2022. doi:10.1371/journal.pcbi.1010045.

24 Katharine Turner, Sayan Mukherjee, and Doug M. Boyer. Persistent homology transform for modeling shapes and surfaces. *Information and Inference: A Journal of the IMA*, 3(4):310–344, 2014.

25 Fan Wang, Hubert Wagner, and Chao Chen. GPU computation of the euler characteristic curve for imaging data. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, volume 224 of *LIPIcs*, pages 64:1–64:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.64.

26 Fan Wang, Hubert Wagner, and Chao Chen. GPU computation of the euler characteristic curve for imaging data. *Journal of Computational Geometry*, 14(2):3–25, December 2023. doi:10.20382/jocg.v14i2a2.

27 Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv preprint, 2017. arXiv:1708.07747.

Near-Optimal Bounds for Parameterized Euclidean k-Means

Vincent Cohen-Addad
Google Research, New York, NY, USA

Karthik C. S.
Rutgers University, Piscataway, NJ, USA

David Saulpic
Université Paris Cité, CNRS, IRIF, F-75013, Paris, France

Chris Schwiegelshohn
Aarhus University, Denmark

Abstract

The k-means problem is a classic objective for modeling clustering in a metric space. Given a set of points in a metric space, the goal is to find k representative points so as to minimize the sum of the squared distances from each point to its closest representative. In this work, we study the approximability of k-means in Euclidean spaces parameterized by the number of clusters, k.

In seminal works, de la Vega, Karpinski, Kenyon, and Rabani [STOC'03] and Kumar, Sabharwal, and Sen [JACM'10] showed how to obtain a $(1+\varepsilon)$-approximation for high-dimensional Euclidean k-means in time $2^{(k/\varepsilon)^{O(1)}} \cdot dn^{O(1)}$.

In this work, we introduce a new fine-grained hypothesis called *Exponential Time for Expanders Hypothesis* (XXH) which roughly asserts that there are no non-trivial exponential time approximation algorithms for the vertex cover problem on near perfect vertex expanders. Assuming XXH, we close the above long line of work on approximating Euclidean k-means by showing that there is no $2^{(k/\varepsilon)^{1-o(1)}} \cdot n^{O(1)}$ time algorithm achieving a $(1+\varepsilon)$-approximation for k-means in Euclidean space. This lower bound is tight as it matches the algorithm given by Feldman, Monemizadeh, and Sohler [SoCG'07] whose runtime is $2^{\widetilde{O}(k/\varepsilon)} + O(ndk)$.

Furthermore, assuming XXH, we show that the seminal $O(n^{kd+1})$ runtime exact algorithm of Inaba, Katoh, and Imai [SoCG'94] for k-means is optimal for small values of k.

2012 ACM Subject Classification Theory of computation → Facility location and clustering; Theory of computation → Computational geometry

Keywords and phrases k-means clustering, Euclidean space, Fine-Grained Complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.33

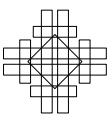
Related Version *Full Version*: https://arxiv.org/abs/2603.28268

Funding *Karthik C. S.*: This work was supported by the National Science Foundation under Grants CCF-2313372 and CCF-2443697, a grant from the Simons Foundation, Grant Number 825876, Awardee Thu D. Nguyen, and partially funded by the Ministry of Education and Science of Bulgaria's support for INSAIT, Sofia University "St. Kliment Ohridski" as part of the Bulgarian National Roadmap for Research Infrastructure.
Chris Schwiegelshohn: This work was partially supported by the Independent Research Fund Denmark (DFF) under a Sapere Aude Research Leader grant No 1051-00106B and by a Google Research Award.

Acknowledgements We thank Dor Minzer, Euiwoong Lee, and Pasin Manurangsi for several discussions that helped conceptualize the proof approach in this paper.



© Vincent Cohen-Addad, Karthik C. S., David Saulpic, and Chris Schwiegelshohn;
licensed under Creative Commons License CC-BY 4.0





1 Introduction

The k-clustering problem represents a fundamental task in data mining and machine learning, providing a model for grouping data points based on similarity. Given a set of points P in a metric space (X, Δ), the objective is to select a set $C \subseteq X$ of k points, referred to as *centers*, and the goal is to minimize an objective function typically defined as the sum of the z-th powers of the distances from each point $p \in P$ to its nearest center in C. This general formulation encompasses several widely studied problems: k-median corresponds to $z = 1$, k-means uses $z = 2$ (minimizing the sum of squared distances), and k-center is when $z \to \infty$ (minimizing the maximum distance).

The algorithmic exploration of k-means, arguably the most popular variant, gained significant traction with the seminal work of Lloyd [49]. Since then, the problem has attracted substantial attention across diverse research communities, including operations research, machine learning, and theoretical computer science.

The computational complexity of k-means problem inherently depends on the structure of the underlying metric space (X, d) and this paper focuses on the setting where points reside in Euclidean space $\mathbb{R}^d$. From a complexity theoretic perspective, Euclidean k-means is known to be NP-Hard even under seemingly restricted conditions, such as when the points lie in the Euclidean plane ($\mathbb{R}^2$) but k is part of the input [58, 51], or when $k = 2$ but the dimension d is large [24, 5]. On the algorithmic side, the best-known exact algorithm for Euclidean k-means was proposed by Inaba, Katoh, and Imai [40], running in time $O(n^{kd+1})$. Remarkably, this has remained the state-of-the-art exact algorithm for over three decades.

Given the hardness of finding exact solutions, particularly in high dimensions, a significant line of research has focused on developing efficient $(1+\varepsilon)$-approximation algorithms. Early breakthroughs by Fernandez de la Vega, Karpinski, Kenyon, and Rabani [30] and Kumar, Sabharwal, and Sen [46] demonstrated that a $(1+\varepsilon)$-approximation for high-dimensional Euclidean k-means could be achieved in time $2^{(k/\varepsilon)^{O(1)}} \cdot dn^{O(1)}$. Subsequent improvements, leveraging techniques such as *coresets* by Feldman, Monemizadeh, and Sohler [29], refined the runtime to $O(ndk) + 2^{\widetilde{O}(k/\varepsilon)}$. Further work by Jaiswal, Kumar, and Sen [42], followed by Jaiswal, Kumar, and Yadav [43], utilized a simpler approach based on D^2*-sampling* to achieve $O(nd \cdot 2^{\widetilde{O}(k/\varepsilon)})$ runtime (see also [1, 10] for a generalized approach).

All these different approaches seem to have hit a running-time barrier at $2^{k/\varepsilon}$ (up to logarithmic factors in the exponent). This contrasts sharply with the related k-center problem, where Agarwal and Procopiuc [4] showed that a $(1+\varepsilon)$-approximation can be obtained much faster, in time $n \log k + (k/\varepsilon)^{O(k^{1-1/d})}$. This runtime for k-center is known to be essentially optimal under the Exponential Time Hypothesis (ETH) [13]. Furthermore, there exists a close relationship between k-means and the Partial Vertex Cover (PVC) problem; indeed, many known hard instances for k-means are derived from hard PVC instances [7, 48, 18]. Interestingly, Manurangsi [53] demonstrated that PVC admits a $(1+\varepsilon)$-approximation in time $\varepsilon^{-O(k)} n^{O(1)}$ (where k is the solution size). This implies that the specific k-means instances derived from PVC can be approximated more efficiently than the runtime of the current best $(1+\varepsilon)$-approximation algorithms for Euclidean k-means.

This discrepancy motivates the central question of our work: When parameterized by the number of clusters k, is the current exponential dependency in k/ε for approximating Euclidean k-means inherent, signifying a fundamental computational gap compared to k-center and PVC-related instances? Or, is it possible to devise a significantly faster approximation algorithm, potentially achieving a runtime closer to those known for related problems and offering a more unified algorithmic picture for k-clustering? More concretely:

▶ **Question 1.** *Is it possible to design a* $(1+\varepsilon)$*-approximation to* k*-means running in time* $O(nd) + \varepsilon^{-O(k)}$ *?*

A negative answer to the above question would also imply progress toward understanding the hardness of exact algorithms. Currently, we are not aware of any fine-grained lower bound that matches the algorithm of Inaba et al. [40]. The closest result we know of is by Cohen-Addad, de Mesmay, Rotenberg, and Roytman [14], who showed that if the centers must be selected from a prescribed set of "candidate centers", then no exact algorithm with a runtime of $n^{o(k)}$ exists for k-median or k-means, even when the dimension is four. It remains an open problem whether a similar result holds for the classic version where centers can be placed anywhere in $\mathbb{R}^d$, and whether an $n^{o(d)}$ lower bound also applies when k is constant.

A negative answer to Question 1 would answer those two questions: modern dimension-reduction [52] and coreset computation [23] reduce the dimension and the number of distinct input points while preserving the clustering cost within a $(1 \pm \varepsilon)$ factor. Thus, as we show in the full version, any lower bound for approximation algorithms translates into a lower bound for exact algorithms, making progress toward demonstrating the optimality of [40].

Negative answers to such questions typically arise from *fine-grained complexity* assumptions, such as the Exponential Time Hypothesis (ETH) [38, 39], or the more modern Gap Exponential Time Hypothesis (Gap-ETH) [55, 26]. Indeed these two assumptions have been very fruitful in explaining the intractability of various important geometric optimization problems [56, 50, 25, 45]. However, Question 1 involves two parameters, k and ε. Suppose we aim to rule out $2^{(k/\varepsilon)^{1-o(1)}} \cdot \mathrm{poly}(n,d)$-time algorithms, then our parameter of interest is the quantity k/ε, where k and ε are free variables constrained only by the fixed ratio k/ε. To the best of our knowledge, such results are not known under ETH or Gap-ETH. In fact, proving such results entail several technical challenges which we discuss in Section 2.1.

1.1 Our Results

In this paper we make substantial progress towards answering these questions. We introduce a hypothesis morally capturing a Gap Exponential Time Hypothesis for Vertex Cover on Near Perfect Vertex Expanders, and we call it *E**x**ponential Time for E**x**panders **H**ypothesis* (denoted **XXH**). This hypothesis is formally stated in Section 3, but an informal discussion about the statement of this hypothesis is given in Section 2.2. Assuming XXH, we are able to answer Question 1 in the following way:

▶ **Theorem 2** (Answer to Question 1; informal statement)**.** *Assuming XXH, for every* $\beta > 0$*, there is no randomized algorithm running in time* $2^{(k/\varepsilon)^{1-\beta}} \cdot \mathrm{poly}(n,d)$ *that can* $(1+\varepsilon)$*-approximate the Euclidean* k*-means problem whenever* $k \gg 1/\varepsilon$*.*

The formal statement of the lower bound is Theorem 12. This result might appear surprising due to the known close connection between vertex cover and k-means. Specifically, a partial vertex cover can be approximated within a $(1+\varepsilon)$ factor in $\varepsilon^{-O(k)} n^{O(1)}$ time [53] (where k is the solution size). This might lead one to expect an approximation scheme for k-means with similar complexity.

For example, reductions such as those in Cohen-Addad et al. [14] and Awasthi et al. [7] (see also [48]) transform the vertex cover instance into a k-means instance by creating a point for each edge of the input graph and positioning the points in space such that pairs of edges sharing a common vertex are close to each other. The hardness proof then relies on distinguishing between two kinds of instances:

(1) Instances derived from graphs admitting a vertex cover of size k, i.e., the edge set can be partitioned to k stars. In this case, the corresponding point set can be partitioned into k clusters such that all points within the same cluster are close (say distance 1, representing edges covered by the same vertex from the cover).

(2) Instances derived from graphs where any set of k vertices leaves at least a constant fraction (say δ fraction) of the edges uncovered. Consequently, in any partitioning of the corresponding point set into k clusters, either a constant fraction of these clusters contain points that are far apart (say distance 3, i.e., representing pairs of edges that do not share one of the selected k vertices) or a few clusters contain a lot of points (and most pairs are far apart).

The hardness for k-means then follows from the difficulty of distinguishing between these two cases based on the clustering cost. Specifically, it involves separating instances admitting a lower cost (e.g., n, associated with Case 1) from those necessitating a higher cost (e.g., $(1-\delta)n + 2\delta n = n + \delta n$, associated with Case 2), where δ is related to the minimum fraction of uncovered edges in the latter case.

However, since partial vertex cover can be approximated within a $(1+\varepsilon)$ factor, for any $\varepsilon > 0$, in time $\varepsilon^{-O(k)}n^{O(1)}$, and since the objective scales linearly with the number of edges not covered (i.e., clients that are at squared distance 3, instead of 1, from their center), this type of instance can be solved in time $\varepsilon^{-O(k)}n^{O(1)}$. Therefore, one cannot expect to boost the lower bound running time from the partial vertex cover result to $2^{(k/\varepsilon)^{1-o(1)}}$.

Thus, to show that the k-means problem requires an exponential dependency in ε, we need to develop a novel reduction framework.

To establish the above conditional lower bound for the Euclidean k-means problem, we first reduce the vertex cover problem in the non-parameterized setting to an intermediate graph problem, essentially in the parameterized setting, but with reduced structure so as to fail standard algorithmic techniques for partial k-vertex cover, while retaining enough structure to embed the graph problem into the Euclidean k-means problem. We direct the reader to Section 2 for further details where we also try to clarify how vertex expansion in the vertex cover problem helps us overcome several technical difficulties.

Theorem 2 implies lower-bounds for exact algorithms as well:

▶ **Corollary 3.** *Assuming XXH, for every $\beta > 0$, there is no algorithm that for any k, d solves the Euclidean k-means problem in time $n^{(k\sqrt{d})^{1-\beta}}$, nor in time $n^{d^{1-\beta}}$.*

Note that this result contrasts the running time of k-means clustering with that of k-center, which admits an exact algorithm running in time $O\left(n^{k^{1-1/d}}\right)$ [4]. The current state of the art algorithm enumerating over all Voronoi partitions in time $O(n^{kd+1})$ by Inaba, Katoh and Imai [40] is thus a likely candidate for being optimal – in particular, it is almost optimal for constant k.

1.2 Further Related Work

Hardness of Clustering. As we mentioned previously, the k-means and k-median problems are NP-hard, even when $k = 2$ (and d is large) [24, 5], or when $d = 2$ (and k is large) [58, 51]. When both parameters are part of the input, the problems become APX-hard [32, 41, 33, 7, 19, 20]. Most techniques to show hardness of approximation are based on reducing from covering problems to clustering problems, for instance through structured instances of max k-coverage or set cover. Recent works have used different approaches: [19] showed how to use hardness of some coloring problems to prove hardness of approximation for k-median and k-means in general metric spaces, and [20] focused on Euclidean spaces and tried to pinpoint what combinatorial structures allow for gap-preserving embeddings to Euclidean space.

For general metrics, the connection between k-clustering and the set cover problem (or rather max-coverage) has been known since the fundamental work of Guha and Khuller [32], who established the best known hardness of approximation bounds. This connection was observed again when analyzing the parameterized complexity of the problem: Cohen-Addad, Gupta, Kumar, Lee and Li [17] showed how to approximate k-median and k-means up to factors $1 + 2/e$ and $1 + 8/e$ respectively and showed that this is tight assuming Gap-ETH (see also [22, 54, 3, 12, 35]).

FPT algorithms via Sketching. The past decades have seen the development of very powerful sketching and compression methods that allow reducing the dimension to $O(\log k/\varepsilon^2)$ [52] and the number of distinct input points to $\widetilde{O}(k\varepsilon^{-z-2})$ ($z = 1$ for k-median, $z = 2$ for k-means) via the construction of coresets [28, 11, 37, 23, 21]. Perhaps surprisingly, these bounds are independent of the original input size and dimension and can be computed in near-linear time, which allows for the construction of simple FPT algorithms. Applying the $O(n^{kd+1})$ algorithm of [40] indeed gives a complexity of $2^{\widetilde{O}(k/\varepsilon^2)}$ plus the near-linear time to sketch the input; naively enumerating all partitions yields a running time of $k^{\widetilde{O}(k/\varepsilon^{z+2})}$, plus the time to sketch the input.

We crucially remark that the dependency on ε cannot be substantially improved: [21] showed a lower bound of $\Omega(k\varepsilon^{-2})$ for coresets, and [47] showed the optimality of the dimension reduction. Therefore, one cannot hope to go below $2^{\Omega(k/\varepsilon^2)}$ and answer Question 1 using only these techniques.

Other parameters were studied for k-means clustering: most notably, the cost has been investigated by Fomin, Golovach and Simonov [31]. They presented a $D^D\text{poly}(nd)$ exact algorithm for k-median, where D is an upper bound on the cost.

From Continuous to Discrete Clustering. The other standard technique to design FPT algorithms is to find a small set of candidate centers that contains a near-optimal solution. This approach was used, for instance, by [8] to obtain the first algorithm running in time $2^{(k/\varepsilon)^{O(1)}}d^{O(1)}n\log^{O(k)}(n)$, by [46] to improve the running time, and by [9] for the capacitated clustering problem.

Approximation Algorithms in Euclidean Spaces. To compute a $(1+\varepsilon)$-approximation in time polynomial in n and k, any algorithm must have a running time at least doubly exponential in d, as the problem is APX-hard in dimension $\Omega(\log n)$. The best of these algorithms is from [16], with a near-linear running time of $f(\varepsilon, d)n\,\text{polylog}\,n$. If one sticks to algorithms polynomial in n, k and d, the lower bounds on the approximation ratio are 1.06 for k-median and 1.015 for k-means, conditioned on P $\neq$ NP [20]. The upper bounds are still quite far: 2.41 for k-median, and 5.96 for k-means [15].

2 Our Techniques

We would like to now convey the conceptual and technical ideas that went into proving the lower bound in Theorem 2. As alluded to earlier in this section, hard instances of Euclidean k-means are typically constructed from the Vertex Cover problem, where every edge is mapped to a client and the partition of the edge set (which is the client set) by an optimal vertex cover also yields the optimal clustering for the k-means objective.

2.1 Motivation and Technical Background

Current Understanding of the Landscape. Starting from Gap-ETH (for 3-SAT) [26, 55], it is easy to show that there is no $2^{o(n)}$ time algorithm to $1+\delta$ approximate the Vertex Cover problem (on sparse graphs) for some small constant $\delta > 0$. By a standard reduction, this implies that Euclidean k-means cannot be approximated within $1+\delta$ in time $2^{o(n)}$, albeit when k is linear in n (the number of clients). On the other hand, starting from ETH [38, 39], it is easy to show (for example following the reduction in [24] or [5]) that Euclidean 2-means cannot be exactly solved in $2^{o(n)}$ time. Thus, we can show that there is no $2^{o(k/\varepsilon)} \cdot \text{poly}(n)$ time algorithm for $1+\varepsilon$ approximating Euclidean k-means problem, when either (i) $k = \Omega(n)$ and $\varepsilon = \Omega(1)$, or (ii) $k = 2$ and $\varepsilon = 1/\Omega(n)$.

Exploring Uncharted Territories. From the above discussion, we know that if k was $\Omega(n)$ then we cannot obtain a $2^{o(n)}$ time approximation algorithm, but what if k was approximately $\sqrt{n}$? Then algorithmic techniques based on coresets yield a $(1+\varepsilon)$ approximation in time $2^{\widetilde{O}(\sqrt{n}/\varepsilon)}$. But is it possible to do better?

We can try to answer this from the lower bound viewpoint. We can look at the above mentioned Gap-ETH hard instance, i.e., the setting $k = \Omega(n)$ and $\varepsilon = \Omega(1)$ and duplicate each point n times to obtain a point-set with $N = n^2$ points, and we have that there is no $2^{\sqrt{N}}$ time algorithm that can constant approximate the objective. But we cannot even rule out the possibility that there is an *exact* algorithm running in time $2^{N^{0.51}}$ for this value of k. Ideally, we would like to be able to rule out algorithms running in time $2^{N^{0.5+\rho}}$ that provide a $(1+\Omega(1/N^{\rho}))$ approximate solution, for every $\rho \in [0, 0.5]$. Thus, the result that we are shooting for is:

$$\textit{Rule out } 2^{o(k/\varepsilon)} \cdot \text{poly}(n) \textit{ time algorithms, when } k/\varepsilon \textit{ is fixed,}$$

i.e., we want the lower bound to hold on the entire tradeoff curve between the number of clusters k and the accuracy of clustering ε. To the best of our knowledge, there are no such results known in fine-grained complexity. Thus, we are motivated to develop a new framework to prove such results.

Technical Challenges. One approach is to start from the lower bound given in (ii), i.e., when $k = 2$ and $\varepsilon = 1/\Omega(n)$, and reduce it to a different instance of k-means, where k has increased (say to $\sqrt{n}$), but also ε has increased to $1/\sqrt{n}$ (all this with a linear blowup in size). But this requires "gap creation", a notorious challenging task, potentially much harder than even proving Gap-ETH from ETH!

Therefore, we pursue the approach of starting from the lower bound given in (i), i.e., when $k = \Omega(n)$ and $\varepsilon = \Omega(1)$, and reduce it to a different instance of k-means, where k has decreased (say to $\sqrt{n}$), and ε has also decreased to $1/\sqrt{n}$. This is the approach of trading off the gap to reduce the number of clusters.

A (Failed) Simple Approach. Let $G = ([n], E)$ be a vertex cover instance (where $|E| := m = O(n)$) which is hard to approximate to $1+\delta$ factor (for some positive constant δ) under Gap-ETH. Thus, size of an optimal vertex cover of G, denoted α_G is $\Omega(n)$. Suppose, our target k-means instance is when $k = o(m)$ and $\varepsilon = \Omega(\delta k/\alpha_G)$: then we may simply look at the embedding where each edge $\{u, v\} \in E$ is mapped to the point (i.e., client), $\mathbf{e}_u + \mathbf{e}_v \in \mathbb{R}^n$, where $\mathbf{e}_i$ is the standard basis vector which is 1 on i^{th} coordinate, and 0 everywhere else. If we were asked to cluster this client set to α_G clusters minimizing the k-means objective in Euclidean metric, then the optimal solution would simply be the partition based on some

optimal vertex cover of G. Thus, we could hope that even when asked to cluster the clients to k parts, the optimal solution would be to first cluster the clients into α_G clusters (based on the vertex cover) and then merge clusters, so as to end up with only k clusters in the end. However, since $k \ll \alpha_G$, we can have near-optimal cost from clustering which do not correspond to any vertex cover of G. For example, a typical obstacle is when we have the following clustering: $k-1$ clusters each contain a single client, and one cluster contains all the remaining points. It is entirely possible that such clusterings also have low cost.

Embedding via Color Coding. To overcome the above issue (of imbalancedness) of clustering, we introduce a color coding based embedding technique. Given G and a target number of clusters k, we first uniformly and independently randomly color each vertex v in G, with a color in $[k]$ (and let color of v be denoted by c_v). Thus each edge (consisting of two vertices), also gets (at most) two colors. Now, consider the embedding where each edge $\{u, v\} \in E$ is mapped to the point (i.e., client), $\mathbf{e}_u + \mathbf{e}_v + \mathbf{e}_{c_u} + \mathbf{e}_{c_v} \in \mathbb{R}^{n+k}$. Consider the clustering of this client set to k parts in the following way. Let $S \subset [n]$ be any optimal vertex cover of G and $S := S_1 \dot\cup \cdots \dot\cup S_k$ be a partition of the vertices in the vertex cover based on the coloring. Then the alleged optimal clusters would be given by the edges covered by $S_1, \ldots, S_k$, i.e., cluster i would be all edges (i.e., the corresponding points of these edges) in G covered by the vertices in S_i (break ties arbitrarily).

The embedding of the colors of the edges, forces the optimal clusters to have a dominating color, and since the colors are uniformly spread, the obstacle mentioned in the above approach (without coloring), would yield high clustering cost and thus can now be ruled out.

That said, there is a more serious obstacle that is not addressed: when we merge clusters in the completeness and soundness case, the graph topology affects the k-means cost; we elaborate on this next.

Need for Vertex Expansion. Consider two graphs G_1 and G_2, both on n vertices and are d-regular. Suppose G_1 looks like a random d regular graph, and thus is a very good vertex expander, and for every subset S of vertices of size $O(n/d)$ we have that the number of unique neighbors is about $|S| \cdot (d - O(1))$ [59]. On the other hand suppose that G_2 is obtained by first taking $n/(2d-6)$ disjoint copies of $K_{d-3,d-3}$ and then adding a random 3-regular graph to connect these copies.

We encounter the following problem when we merge clusters as described above: G_1 behaves in the way we expect it to, whereas the G_2 has a very low cost even if it doesn't have a good vertex cover. This is because if we take the $d-3$ sized independent set in a $K_{d-3,d-3}$ then we do get a very good vertex cover of $K_{d-3,d-3}$ but in addition if all (or even a large fraction of) the edges of $K_{d-3,d-3}$ were in the same cluster then each edge has $2d-8$ other edges that is adjacent to it in the cluster. On the other hand, if we take the edges incident to an independent set in G_1, then a typical edge has $d-1$ other edges incident to it. This makes the analysis of completeness and soundness impossible without knowing more about the graph topology.

2.2 New Hypothesis: Exponential Time for Expanders Hypothesis

Exponential Time for Expanders Hypothesis (XXH). To remedy the situation we introduce a working hypothesis that the gap vertex cover problem cannot be solved in time $2^{n^{1-o(1)}}$ on random graphs. For the sake of keeping the proofs clean (to the extent possible), we state the hypothesis in Section 3 in terms of vertex expanders which makes it directly usable. Under this hypothesis, we use the color coding embedding that was described earlier and with a lot of technical effort are able to derive the conclusion given in Theorem 2.

Informally, XXH asserts that for some constants ζ, δ, such that $\zeta \ll \delta$, no randomized algorithm running in $2^{n^{1-o(1)}}$ time can take as input a d-regular vertex expander $G := ([n], E)$ and distinguish between the following:
Completeness: There exist $n/2$ vertices that cover at least $(1-\zeta)$ fraction of E.
Soundness: Every subset of V of size $n/2$ does not cover δ fraction of the edges.

Plausibility of XXH. A key observation connecting random graphs and expander graphs is that random d-regular graphs are known to be very strong vertex expanders with constant probability (see Theorem 4.2 in [59]). Consequently, making progress on understanding the inapproximability of the Vertex Cover problem on vertex expanders is closely related to the hardness of the vertex cover problem on random graphs.

XXH formalizes this hardness on vertex expanders and, via our main reduction (Theorem 12), links it directly to the Euclidean k-means problem. This connection implies we live in one of three possible worlds. The first is where XXH is false and the Euclidean k-means admits a much faster algorithm (such as $2^{(k/\varepsilon)^{0.99}} \cdot \mathrm{poly}(n)$ time), and this would yield new algorithms for computing Vertex Cover on random-like graphs. The second is where XXH is false but there are no new faster algorithms for Euclidean k-means, and this would force a deeper, computational understanding of vertex expanders themselves. The third world is where XXH is true, and thus the current Euclidean k-means algorithm is nearly optimal. The reader is directed to Section 3.2 for more discussion on these three possibilities.

Investigating the truth of XXH (the third world) leads to natural questions of independent interest. For instance, Ramanujan graphs are known to be near-perfect vertex expanders for small sets [57], so we ask: does the Vertex Cover problem admit a PTAS when the input graph is a Ramanujan graph? Progress on this question would directly help us better understand XXH.

As partial evidence for XXH, in Corollary 11 we show that if we forego the expansion property, then under the Unique Games Conjecture it is possible to show from [6] that no polynomial-time algorithm can take as input a d-regular graph $G := ([n], E)$ and distinguish between the following:
Completeness: There exist $n/2$ vertices that cover at least a $(1-\zeta)$ fraction of E.
Soundness: Every subset of V of size $n/2$ fails to cover an $\Omega(\sqrt{\zeta})$ fraction of the edges.

Our Message. We wish to highlight two key aspects of the XXH hypothesis. First, it fundamentally acts as a fine-grained complexity assumption regarding the hardness of the Vertex Cover problem on random instances. Second, it establishes an important connection: developing better approximation algorithms for Euclidean k-means would provide non-trivial approaches to solving Vertex Cover on expander graphs, a problem of significant independent interest. In addition, even if weaker versions of XXH are true (and proved in the future), this would imply weaker lower bounds for $(1+\varepsilon)$-approximating the Euclidean k-means problem (see Remark 14 for details).

2.3 Proof Overview of Theorem 2

We now give an overview of the completeness and soundness cases. Recall that we are given a d-regular graph $G = ([n], E)$, and we have constructed a point $\mathbf{e}_u + \mathbf{e}_v + \mathbf{e}_{c_u} + \mathbf{e}_{c_v} \in \{0,1\}^{n+k}$, for each edge (u, v) whose end points have colors c_u and c_v.

In the completeness case, we have that there are $n/2$ vertices cover $1-\zeta$ fraction of edges, so our strategy to cluster is straightforward: first form $n/2$ clusters, each one corresponding to a vertex in the vertex cover solution, and then identify each cluster with a color by looking

at the color of the common vertex in each cluster (which is a star graph). Then, each color class would be a cluster, and we have k clusters, and we can show that the cost is $3|E| - (1 - 7\zeta)kd$.

However, our soundness analysis is highly non-trivial, and involves several tools and arguments of different flavors. Suppose we have a clustering of P whose k-means cost is about $3|E| - (1 - \delta^5)kd$ (where $\zeta \leq \delta^5/10$), then we first connect the cost of cluster C_i (for $i \in [k]$) with certain properties of the graph G_i in the following way:

▶ **Lemma 4** (Informal statement). *For every cluster C_i we have its cost is equal to:*

$$3|C_i| - 1 + (\gamma_i - \kappa_i)(|C_i| - 1) - \frac{1}{|C_i|} \cdot \sum_{v \in V_i} d_{i,v}^2,$$

where $d_{i,v}$ is the degree of v in the graph G_i (induced by edges in C_i), γ_i is the fraction of pairs of edges in G_i that have no color in common, and κ_i is the fraction of pairs of edges that have two colors in common.

Next, we show that if γ_i (fraction of pairs of edges with no color in common) and κ_i (fraction of pairs of edges with two colors in common) are not too large, then there is a dominant single color in the cluster C_i.

▶ **Lemma 5** (Informal statement). *If κ_i and γ_i are bounded by some small constants, then there is a large fraction of edges in G_i that have the same color.*

Then, we show that we can identify a large subcollection of clusters for which both κ_i and γ_i are small. In addition, we assume that the cost of the clustering $C_1, \dots, C_k$ is at most $3|E| - (1 - \delta^5)dk$. Then, for each cluster in this subcollection, we can relate a bound on the sum of the squared degrees of the vertices in the subcollection to the clustering cost appearing in Lemma 4.

▶ **Lemma 6** (Informal statement). *There is some $I \subseteq [k]$ such that for all $i \in I$, it holds that κ_i and γ_i are small and $\sum_{i \in I} |E_i| \geq (1 - 38\delta^2) \cdot |E|$. In addition, we also have for all $i \in I$:*

$$\frac{\sum_{v \in G_i} d_{i,v}^2}{\sum_{v \in G_i} d_{i,v}} \geq (1 - \delta^3) \cdot \frac{d}{2}.$$

Finally, we show that a cost of clustering $3|E| - (1 - \delta^5)dk$ implies that we can construct a set of vertices $S \subseteq [n]$ of size slightly more than $n/2$ such that we cover $(1 - 10\delta^{1.5})$ fraction of the edges, contradicting the soundness assumption of XXH for small enough δ. This step is quite involved, and we skip providing more details about it here.

3 Exponential Time for Expanders Hypothesis

In this section, we formally introduce the Exponential Time for Expanders Hypothesis which is then used in the next section to prove conditional lower bounds for the Euclidean k-means problems for small k.

3.1 New Hypothesis: Exponential Time for Expanders Hypothesis

We first define the notion of vertex expansion relevant to this submission.

▶ **Definition 7** (Small Set Vertex Expanders). *Given constants $\alpha > 0$, a d-regular graph $G = (V, E)$ on n vertices, and an integer $k := k(n)$, we say that G is a (k, α)-small set vertex expander if for every subset $S \subseteq V$ of size at most n/k we have that $|\{u \in V : \exists\{u, v\} \in E \text{ and } v \in S\}| \geq (1 - \alpha) \cdot d \cdot |S|$.*

Now, we can define our new hypothesis.

▶ **Hypothesis 8** (Exponential Time for Expanders Hypothesis – XXH(δ, ζ, α)). *Given constants $\delta, \zeta, \alpha \in (0, 1)$, the XXH$(\delta, \zeta, \alpha)$ assumption states that the following holds for all $\beta > 0$: No randomized algorithm running in $2^{n^{1-\beta}}$ time can, given as input a d-regular $(\text{polylog}n, \alpha)$-small set vertex expander $G = (V = [n], E)$ with $d = (\log n)^L$ (for some $L > 1$), distinguish between the following with probability 0.9:*

Completeness: *There exist $n/2$ vertices that cover at least $(1 - \zeta)$ fraction of E.*

Soundness: *For every $S \subseteq V$ of size $n/2$, there are at least $\delta \cdot |E|$ many edges which are not covered by any vertex in S.*

3.2 Plausibility of XXH: Three Possible Worlds

For the sake of the discussion in this subsection, we refer to the event "Clustering Barrier is Breached" to simply denote the existence of an algorithm much faster than $2^{k/\varepsilon} \cdot \text{poly}(n)$ for $1 + \varepsilon$ approximating the Euclidean k-means problem. In Section 4, we proved that assuming XXH, the clustering barrier cannot be breached. Therefore, we are living in one of three possible worlds. The first world is where XXH is false and the clustering barrier is breached. The second world is where XXH is false, but the clustering barrier cannot be breached. Finally, the third world is where XXH is true (and thus the clustering barrier cannot be breached from Theorem 12).

The message we want to convey here is that only one of the above three worlds is possible, and regardless of which world is proven to be the one we live in, it will shed new light on problems of interest to the algorithmic community.

3.2.1 World I: XXH is False and Clustering Barrier is Breached

XXH may be viewed as a fine-grained assumption for the vertex cover problem on random instances, which we elaborate on below. Thus, our reduction from XXH instances to Euclidean k-means instances, as given in Theorem 12, can be used to make oracle calls to the efficient algorithm for the Euclidean k-means problem (since the clustering barrier is breached in this world) to solve XXH instances efficiently (i.e., in mildly sub-exponential time). This would imply a separation in this world between worst-case and average-case instances of the gap vertex cover problem (brushing aside many details to make a succinct claim).

XXH as a Fine-Grained Assumption for Random Instances. Theorem 4.2 in [59] shows that for some large universal constant C, a random d-regular graph is a $(C, \frac{2}{d})$-small set vertex expander with a probability of 0.5. Thus, making progress on the inapproximability of the vertex cover problem for vertex expanders is morally similar to proving the hardness of approximation for vertex cover on random d-regular graphs. Or in this world, we would make algorithmic progress on understanding vertex cover on random graphs through (hypothetical) clustering algorithms.

3.2.2 World II: XXH is False but Clustering Barrier is not Breached

In this world, XXH is false, possibly because[1] the vertex cover problem is computationally easy on small set vertex expanders. For *spectral* expanders, it is sometimes possible to apply Hoffman's ratio bound [34] to obtain non-trivial speedups when the spectral gap is large. However, vertex expanders, while intuitively similar to edge expanders (and spectral ones), are poorly understood. In fact, explicit constructions with parameters close to those obtained for random graphs were developed only recently [36]. Thus, falsifying XXH motivates a better understanding of vertex expanders from the computational viewpoint of optimization problems.

Nevertheless, Theorem 11 in [2] shows that the *exact* vertex cover problem remains as hard on spectral expanders as it is on general graphs.

3.2.3 World III: XXH is True

The first evidence that XXH might be true is that the hard instances of vertex cover constructed by [27] are essentially built from random label cover instances which have strong expansion properties after PCP composition. Moreover, XXH only promises expansion for small sets (sub-polynomial size sets), and thus it is unlikely that such a local structure can be algorithmically used.

Also, it can be formally argued that on small sets, Ramanujan graphs are near-perfect vertex expanders [57]. Therefore, as a way to prove XXH, one can first ask whether the vertex cover problem admits a PTAS on Ramanujan graphs. It is possible that the answer to this question is negative, although there are no techniques to handle such questions, and thus XXH opens this new line of exploration.

3.3 Small Progress on XXH under Unique Games Conjecture

In this subsection, we show that without the vertex expansion property, it is possible to obtain a weak version of XXH under the unique games conjecture [44].

Let Φ denote the cumulative density function of the standard normal distribution and, for any $\rho \in [-1,1], \mu \in [0,1]$, let $\Gamma_\rho(\mu)$ denote $\Pr[X \leq \Phi^{-1}(\mu) \wedge Y \leq \Phi^{-1}(\mu)]$ where X, Y are normal random variables with means 0, variances 1 and covariance ρ. The main intermediate result of [6] is the following:

▶ **Theorem 9** (Theorem 1 from [6]). *For any $q \in (0, 1/2)$ and any $\varepsilon > 0$, it is UG-hard to, given a regular graph $G = (V, E)$, distinguish between the following two cases.*

- *(Completeness) G contains an independent set of size at least $q \cdot |V|$.*
- *(Soundness) For any subset $T \subseteq V$, the number of edges with both endpoint in T is at least $|E| \cdot \left(\Gamma_{-q/(1-q)}(\mu) - \varepsilon\right)$ where $\mu = |T|/|V|$.*

This means that, in the completeness case, there is a vertex cover of size at most $(1-q)\cdot|V|$. On the other hand, in the soundness case, if we consider any subset $S \subseteq V$ of size at most $(1-q)\cdot|V|$, then the number of edges *not* covered is exactly the same as the number of edges with both endpoints in $(V \setminus S)$, which is at least $(\Gamma_{-q/(1-q)}(q) - \varepsilon)\cdot|E|$. We will evaluate this approximation factor for $q = \frac{1}{2} - \zeta$.

[1] We are not addressing here the concern that XXH might be false in this world because of the setting of the parameters.

▷ Claim 10. Let $q := \frac{1}{2} - \zeta$ where $\zeta > 0$ is small enough. We have: $\Gamma_{-q/(1-q)}(q) \geq \frac{\sqrt{\zeta}}{3}$.

Assuming the above claim (which will be proved later), we have the following corollary from Theorem 9 by setting $\varepsilon \ll \sqrt{\zeta}$:

▶ **Corollary 11.** *There exists $\delta, \zeta > 0$ with $\delta^2 > \zeta/33$ such that it is UG-hard to, given a regular graph $G = (V, E)$, distinguish between the following two cases.*
Completeness: *There exist $|V|/2$ vertices that cover at least $(1 - \zeta)$ fraction of E.*
Soundness: *For every $S \subseteq V$ of size $|V|/2$, there are at least $\delta \cdot |E|$ many edges which are not covered by any vertex in S.*

This corollary is weaker than the promise given in XXH in the following ways. First, and most importantly, the hard instances of Corollary 11 need not be vertex expanders. Second, the conditional lower bound under the Unique Games Conjecture is only against polynomial-time algorithms, whereas XXH rules out sub-exponential time algorithms.

Between the two remarks made above, the first one is the major obstacle, and this is mainly to do with our state of poor understanding of vertex expanders. Once the toolkit develops on this topic, it is conceivable that some additional progress can be made on XXH.

4 Conditional Lower Bound for Euclidean k-means with Few Clusters

We prove in this section our main result, the hardness of approximating k-means when parameterized by k. The formal theorem is:

▶ **Theorem 12** (Fine-Grained Hardness for Approximation of Euclidean k-means from XXH)**.** *Suppose XXH(δ, ζ, α) is true for some constants $\delta, \zeta, \alpha \in (0, 1)$ satisfying $\delta \leq 10^{-3}$, $\zeta \leq \delta^5/10$, and $\alpha \leq \delta^{10}$. Let $L > 1$ be the constant defining the degree $d = (\log |V|)^L$ in the XXH hypothesis.*

Let $\widetilde{k} : \mathbb{N} \to \mathbb{N}$ be a non-decreasing function and $\widetilde{\varepsilon} : \mathbb{N} \to (0, 1)$ be a non-increasing function. Define $f(n) := \frac{\widetilde{k}(n)}{\widetilde{\varepsilon}(n)}$. Assume that for all sufficiently large n, $f(n)$ satisfies the smoothness condition $f(n) \leq C \cdot f(n-1)$ for some constant $C \geq 1$. Furthermore, assume the following asymptotic limits hold:

- $\widetilde{k}(1) \geq \frac{1}{\widetilde{\varepsilon}(1)} \geq 2/\delta^5$.
- $f(n) = \omega(\log n)$ *and* $f(n) = o\left(\frac{n}{\log^L n}\right)$.
- $\widetilde{\varepsilon}(n) = o\left(\frac{1}{(\log n)^{\omega(1)}}\right)$.
- $\widetilde{k}(n) \cdot \widetilde{\varepsilon}(n) = \omega\left((\log n)^{2L}\right)$.

Then the following holds for all $\beta > 0$: No randomized algorithm can, given as input exactly n points in $\mathbb{R}^{\text{poly}(n)}$ and an integer $\widetilde{k}(n)$, run in $2^{f(n)^{1-\beta}} \cdot \text{poly}(n)$ time and output a $(1 + \widetilde{\varepsilon}(n))$-approximate estimate of the $\widetilde{k}(n)$-means cost with probability at least 0.9.

In this section, we actually prove the following theorem, which applies for $\widetilde{k}$ and $\widetilde{\varepsilon}$ bounded in a specific way and then show that it immediately implies Theorem 12 above.

▶ **Theorem 13.** *Let $\delta, \zeta, \alpha \in (0, 1)$ be some constants satisfying $\delta \leq 10^{-3}$, $\zeta \leq \delta^5/10$, and $\alpha \leq \delta^{10}$. There is a randomized algorithm running in linear time in the input size, which takes as input an integer k (where $k > \sqrt{|V|} \cdot d$, $k = o(|V|/d^{\omega(1)})$) and a d-regular $(\text{polylog}|V|, \alpha)$-small set vertex expander $G = (V, E)$ (where $d = (\log |V|)^L$, for some $L > 1$), and outputs a point-set $P \subseteq \mathbb{R}^{O(|V|)}$ of at most $|E|$ points such that with probability at least 0.95, the following holds:*

Completeness: *If there are $|V|/2$ vertices covering at least $1-\zeta$ fraction of E, then there is a clustering of P such that the k-means cost is at most $3|E| - (1-7\zeta)kd$.*

Soundness: *If every $|V|/2$ vertices miss at least δ fraction of E, then every clustering of P has k-means cost at least $3|E| - (1-\delta^5)kd$.*

▶ Remark 14. It is worth noting that the reduction in Theorem 13 applies even for weaker versions of XXH. For example, if one day in the future, XXH was proved against algorithms running in time $2^{\sqrt{n}}$ instead of the currently stated $2^{n^{1-o(1)}}$ runtime algorithms, then we can recover that there is no randomized algorithm running in time much faster than $2^{\sqrt{k/\varepsilon}} \cdot \mathrm{poly}(n,d)$ that can $(1+\varepsilon)$-approximate the Euclidean k-means problem.

References

1 Fateme Abbasi, Sandip Banerjee, Jaroslaw Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. Parameterized approximation schemes for clustering with general norm objectives. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1377–1399. IEEE, 2023. doi:10.1109/FOCS57990.2023.00085.

2 Amir Abboud and Nathan Wallheimer. Worst-case to expander-case reductions. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 1:1–1:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.1.

3 Marek Adamczyk, Jaroslaw Byrka, Jan Marcinkowski, Syed Mohammad Meesum, and Michal Wlodarczyk. Constant-factor FPT approximation for capacitated k-median. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 1:1–1:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.1.

4 Pankaj K Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33:201–226, 2002. doi:10.1007/S00453-001-0110-Y.

5 Enver Aman, Karthik C. S., and Sharath Punna. On connections between k-coloring and euclidean k-means. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 9:1–9:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.9.

6 Per Austrin, Subhash Khot, and Muli Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. *Theory of Computing*, 7(1):27–43, 2011. doi:10.4086/TOC.2011.V007A003.

7 Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of euclidean k-means. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, volume 34 of *LIPIcs*, pages 754–767. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.SOCG.2015.754.

8 Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 250–257. ACM, 2002. doi:10.1145/509907.509947.

9 Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Faster algorithms for the constrained k-means problem. *Theory Comput. Syst.*, 62(1):93–115, 2018. doi:10.1007/s00224-017-9820-7.

10 Sujoy Bhore, Ameet Gadekar, and Tanmay Inamdar. Coreset strikes back: Improved parameterized approximation schemes for (constrained) k-median/means, 2025. doi:10.48550/arXiv.2504.06980.

11 Vladimir Braverman, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for clustering in excluded-minor graphs and beyond. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2679–2696. SIAM, 2021. Consulted on arxiv on May 2022. doi:10.1137/1.9781611976465.159.

12 Xianrun Chen, Lu Han, Dachuan Xu, Yicheng Xu, and Yong Zhang. k-median/means with outliers revisited: A simple fpt approximation. In Weili Wu and Guangmo Tong, editors, *Computing and Combinatorics - 29th International Conference, COCOON 2023, Hawaii, HI, USA, December 15-17, 2023, Proceedings, Part II*, volume 14423 of *Lecture Notes in Computer Science*, pages 295–302. Springer, 2023. doi:10.1007/978-3-031-49193-1_22.

13 Rajesh Chitnis and Nitin Saurabh. Tight lower bounds for approximate & exact k-center in ℓ_∞^d. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, volume 224 of *LIPIcs*, pages 28:1–28:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SOCG.2022.28.

14 Vincent Cohen-Addad, Arnaud de Mesmay, Eva Rotenberg, and Alan Roytman. The bane of low-dimensionality clustering. In *Proceedings of the 2018 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2018. doi:10.1137/1.9781611975031.30.

15 Vincent Cohen-Addad, Hossein Esfandiari, Vahab S. Mirrokni, and Shyam Narayanan. Improved approximations for euclidean k-means and k-median, via nested quasi-independent sets. In *STOC '22*. ACM, 2022. doi:10.1145/3519935.3520011.

16 Vincent Cohen-Addad, Andreas Emil Feldmann, and David Saulpic. Near-linear time approximation schemes for clustering in doubling metrics. *J. ACM*, 68(6):44:1–44:34, 2021. doi:10.1145/3477541.

17 Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT approximations for k-median and k-means. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.42.

18 Vincent Cohen-Addad and Karthik C. S. Inapproximability of clustering in lp metrics. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 519–539. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00040.

19 Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. On approximability of clustering problems without candidate centers. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2635–2648. SIAM, 2021. doi:10.1137/1.9781611976465.156.

20 Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. Johnson coverage hypothesis: Inapproximability of k-means and k-median in ℓ_p-metrics. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1493–1530. SIAM, 2022. doi:10.1137/1.9781611977073.63.

21 Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, and Chris Schwiegelshohn. Towards optimal lower bounds for k-median and k-means coresets. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1038–1051. ACM, 2022. doi:10.1145/3519935.3519946.

22 Vincent Cohen-Addad and Jason Li. On the fixed-parameter tractability of capacitated clustering. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 41:1–41:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.41.

23 Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 169–182. ACM, 2021. doi:10.1145/3406325.3451022.

24 Sanjoy Dasgupta and Yoav Freund. Random projection trees for vector quantization. *IEEE Transactions on Information Theory*, 55(7):3229–3242, 2009. doi:10.1109/TIT.2009.2021326.

25 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for exponential-time-hypothesis-tight algorithms and lower bounds in geometric intersection graphs. *SIAM J. Comput.*, 49(6):1291–1331, 2020. doi:10.1137/20M1320870.

26 Irit Dinur. Mildly exponential reduction from gap 3sat to polynomial-gap label-cover. *Electron. Colloquium Comput. Complex.*, TR16-128, 2016. URL: https://eccc.weizmann.ac.il/report/2016/128.

27 Irit Dinur and Shmuel Safra. The importance of being biased. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 33–42. ACM, 2002. doi:10.1145/509907.509915.

28 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 569–578, 2011. doi:10.1145/1993636.1993712.

29 Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A PTAS for k-means clustering based on weak coresets. In Jeff Erickson, editor, *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, pages 11–18. ACM, 2007. doi:10.1145/1247069.1247072.

30 Wenceslas Fernandez de la Vega, Marek Karpinski, Claire Kenyon, and Yuval Rabani. Approximation schemes for clustering problems. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 50–58. ACM, 2003. doi:10.1145/780542.780550.

31 Fedor V. Fomin, Petr A. Golovach, and Kirill Simonov. Parameterized k-clustering: Tractability island. *J. Comput. Syst. Sci.*, 117:50–74, 2021. doi:10.1016/J.JCSS.2020.10.005.

32 Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *J. Algorithms*, 31(1):228–248, 1999. doi:10.1006/jagm.1998.0993.

33 Venkatesan Guruswami and Piotr Indyk. Embeddings and non-approximability of geometric problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 537–538, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644198.

34 Willem H Haemers. Hoffman's ratio bound. *Linear Algebra and its Applications*, 617:215–219, 2021.

35 Lu Han, Dachuan Xu, Donglei Du, and Dongmei Zhang. An approximation algorithm for the uniform capacitated k-means problem. *J. Comb. Optim.*, 44(3):1812–1823, 2022. doi:10.1007/S10878-020-00550-Y.

36 Jun-Ting Hsieh, Alexander Lubotzky, Sidhanth Mohanty, Assaf Reiner, and Rachel Yun Zhang. Explicit lossless vertex expanders. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia*. IEEE, 2025.

37 Lingxiao Huang and Nisheeth K. Vishnoi. Coresets for clustering in euclidean spaces: importance sampling is nearly optimal. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1416–1429. ACM, 2020. doi:10.1145/3357713.3384296.

38 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

39 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

40 Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based *k*-clustering (extended abstract). In *Proceedings of the Tenth Annual Symposium on Computational Geometry, Stony Brook, New York, USA, June 6-8, 1994*, pages 332–339, 1994. doi:10.1145/177424.178042.

41 Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 731–740, 2002. doi:10.1145/509907.510012.

42 Ragesh Jaiswal, Amit Kumar, and Sandeep Sen. A simple D 2-sampling based PTAS for k-means and other clustering problems. *Algorithmica*, 70(1):22–46, 2014. doi:10.1007/S00453-013-9833-9.

43 Ragesh Jaiswal, Mehul Kumar, and Pulkit Yadav. Improved analysis of d2-sampling based ptas for k-means and other clustering problems. *Information Processing Letters*, 115(2):100–103, 2015. doi:10.1016/J.IPL.2014.07.009.

44 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775, 2002. doi:10.1145/509907.510017.

45 Sándor Kisfaludi-Bak, Jesper Nederlof, and Karol Wegrzycki. A gap-eth-tight approximation scheme for euclidean TSP. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 351–362. IEEE, 2021. doi:10.1109/FOCS52979.2021.00043.

46 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2), 2010. doi:10.1145/1667053.1667054.

47 Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 633–638. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.64.

48 Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Inf. Process. Lett.*, 120:40–43, 2017. doi:10.1016/j.ipl.2016.11.009.

49 Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. doi:10.1109/TIT.1982.1056489.

50 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bull. EATCS*, 105:41–72, 2011. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/92.

51 Meena Mahajan, Prajakta Nimbhorkar, and Kasturi R. Varadarajan. The planar k-means problem is np-hard. *Theor. Comput. Sci.*, 442:13–21, 2012. doi:10.1016/J.TCS.2010.05.034.

52 Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Performance of johnson-lindenstrauss transform for *k*-means and *k*-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1027–1038, 2019. doi:10.1145/3313276.3316350.

53 Pasin Manurangsi. A note on max k-vertex cover: Faster fpt-as, smaller approximate kernel and improved approximation. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASIcs*, pages 15:1–15:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/OASIcs.SOSA.2019.15.

54 Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of maximum k-coverage, unique set cover and related problems (via t-wise agreement testing theorem). In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 62–81. SIAM, 2020. doi:10.1137/1.9781611975994.5.

55 Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense csps. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 78:1–78:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.78.

56 Dániel Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008. doi:10.1093/COMJNL/BXM048.

57 Theo McKenzie and Sidhanth Mohanty. High-Girth Near-Ramanujan Graphs with Lossy Vertex Expansion. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 96:1–96:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.96.

58 Nimrod Megiddo and Kenneth J Supowit. On the complexity of some common geometric location problems. *SIAM journal on computing*, 13(1):182–196, 1984. doi:10.1137/0213014.

59 Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

Almost-Optimal Upper and Lower Bounds for Clustering in Low Dimensional Euclidean Spaces

Vincent Cohen-Addad
Google Research, New York, NY, USA

Karthik C. S.
Rutgers University, Piscataway, NJ, USA

David Saulpic
Université Paris Cité, CNRS, IRIF, F-75013, Paris, France

Chris Schwiegelshohn
Aarhus University, Denmark

Abstract

The k-median and k-means clustering objectives are classic objectives for modeling clustering in a metric space. Given a set of points in a metric space, the goal of the k-median (resp. k-means) problem is to find k representative points so as to minimize the sum of the distances (resp. sum of squared distances) from each point to its closest representative. Cohen-Addad, Feldmann, and Saulpic [JACM'21] showed how to obtain a $(1+\varepsilon)$-factor approximation in low-dimensional Euclidean metric for both the k-median and k-means problems in near-linear time $2^{(1/\varepsilon)^{O(d^2)}} n \cdot \text{polylog}(n)$ (where d is the dimension and n is the number of input points).

We improve this running time to $2^{\widetilde{O}(1/\varepsilon)^{d-1}} \cdot n \cdot \text{polylog}(n)$, and show an almost matching lower bound: under the Gap Exponential Time Hypothesis for 3-SAT, there is no $2^{o(1/\varepsilon^{d-1})} n^{O(1)}$ algorithm achieving a $(1+\varepsilon)$-approximation for k-means.

2012 ACM Subject Classification Theory of computation → Facility location and clustering; Theory of computation → Computational geometry

Keywords and phrases k-means clustering, k-median clustering, Euclidean space, Fine-Grained Complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.34

Related Version *Full Version*: https://arxiv.org/abs/2603.09846

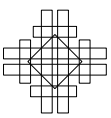
Funding *Karthik C. S.*: This work was supported by the National Science Foundation under Grants CCF-2313372 and CCF-2443697, a grant from the Simons Foundation, Grant Number 825876, Awardee Thu D. Nguyen, and partially funded by the Ministry of Education and Science of Bulgaria's support for INSAIT, Sofia University "St. Kliment Ohridski" as part of the Bulgarian National Roadmap for Research Infrastructure.
Chris Schwiegelshohn: This work was partially supported by the Independent Research Fund Denmark (DFF) under a Sapere Aude Research Leader grant No 1051-00106B and by a Google Research Award.

1 Introduction

The k-median and k-means problems are minimization tasks of classic objectives for modeling clustering in a wide variety of applications arising in data mining and machine learning. Given a set of points $P \subset X$ and a set of *candidate centers* $\mathcal{S} \subset X$ in a metric space (X, Δ),



© Vincent Cohen-Addad, Karthik C. S., David Saulpic, and Chris Schwiegelshohn;
licensed under Creative Commons License CC-BY 4.0



the goal of the k-median problem[1] is to find a set of k points $S \subset \mathcal{S}$, called *centers*, so as to minimize the sum of the distances (given by Δ) from each point of P to its closest center in S (in the k-means problem, the goal is to instead minimize the sum of the squared distances from each point of P to its closest center in S).

The algorithmic study of k-median started in the early '60s in the operations research community [36, 46, 29], while the study of k-means was arguably initiated by the seminal work of Lloyd [39], which applied clustering to quantization of analog signals. Since then, both problems have received a tremendous amount of attention. Naturally, the complexity of the problems varies with the underlying metric space hosting the input points. On the negative side, both problems are known to be NP-hard in Euclidean space, even when the points lie in the Euclidean plane (and k is large) [44, 40], or when $k = 2$ [23, 2] (and the dimension is large). When both k and d are arbitrary, several groups of researchers have shown hardness of approximation results for both problems [6, 37, 17, 19]. Even stronger hardness results hold if the input lives in an arbitrary metric space (see Guha and Kuller [28] and [18]).

Fine-Grained Complexity. Despite those hardness results, the importance of Euclidean inputs in statistics and machine learning applications has led researchers to develop methods and algorithms, in particular through the study of the parameterized complexity of the problem, when the input lies in an Euclidean space. Both the dimensionality of the input, d, and the target number of clusters, k, have been studied as parameters. For exact algorithms, the seminal work of Inaba, Katoh and Imai [31] has shown that one can compute an exact solution to the k-median and k-means problems in time $O(n^{kd+1})$ – as opposed to the naive brute-force enumeration which runs in $\widetilde{O}(k^n)$ time. On the negative side, Cohen-Addad, de Mesmay, Rotenberg and Roytman [11] much later showed that there is no $n^{o(k)}$ exact algorithm for (discrete) k-median or k-means problem, even when the dimension is four, and thus partially showing the optimality of the $O(n^{kd+1})$ algorithm.

To circumvent this hardness, researchers have turned to approximation algorithms. When the dimension d and the precision ε are taken as parameters, a line of work [5, 35, 27, 20, 10, 13] improved the running time of approximation schemes from quasi-polynomial time, $2^{(\log(n)/\varepsilon)^{d-1}}$, to $2^{(1/\varepsilon)^{O(d^2)}}n$. The doubly exponential dependency in d^2 stands in stark contrast with other similar results, e.g., for the Traveling Salesperson Problem (TSP). For this problem, Kisfaludi-Bak, Nederlof and Wegrzycki [34] recently settled the exact dependency on d, by showing an algorithm running essentially in time $2^{O\left(1/\varepsilon^{d-1}\right)}n$, and a lower bound of $2^{\Omega\left((1/\varepsilon)^{d-1}\right)}$ under the Gap-ETH hypothesis [25, 42]. In light of this result, we consider the following question:

▶ **Question 1.** *Is it possible to obtain a $2^{O\left(1/\varepsilon^{d-1}\right)}n^{O(1)}$ approximation scheme for k-median and k-means problems in low-dimensional inputs? Would it be possible to get an even faster $2^{o\left(1/\varepsilon^{d-1}\right)}n^{O(1)}$ approximation scheme for both problems?*

1.1 Our Results

In this paper, we make substantial progress towards answering Question 1. Our first contribution is the design of an improved algorithm:

▶ **Theorem 2.** *For every $\varepsilon > 0$ and dimension d, the k-median and k-means problems in $\mathbb{R}^d$ can both be approximated to a $(1+\varepsilon)$-factor in time $2^{\widetilde{O}\left(1/\varepsilon^{d-1}\right)}n \cdot \mathrm{polylog}(n)$.*[2]

[1] This variant is sometimes referred to as the *discrete* k-median problem. In the continuous k-median problem we are allowed to pick the k centers S from anywhere in X.

[2] The $\widetilde{O}$ notation hides an exponential dependency in d, and polynomial dependency in $\log(1/\varepsilon)$.

The above algorithm is our main contribution, and improves on the techniques of [13]. It also readily extends to the continuous k-median and k-means problems (see Remark 4).

On the technical side, this algorithm deepens our understanding of the *quadtree* decomposition for clustering. A long line of work based on this decomposition [5, 35, 27, 20, 10, 13] improved the running time of approximation schemes (i.e., $(1+\varepsilon)$-factor approximation algorithms for any small enough constant ε) for clustering from quasi-polynomial time, $2^{(\log(n)/\varepsilon)^{d-1}}$, to $2^{(1/\varepsilon)^{O(d^2)}}n$ [13]. The main conceptual insight of these works is to bound the number of *portals* necessary along the decomposition so that forcing a path to go through portals incurs at most a $(1+\varepsilon)$-factor distortion. The main contribution of [13] is that after processing a tiny fraction of the input, only a constant number of portals are sufficient at each level of the quadtree decomposition. However, this constant remains much larger than that required for other natural problems.

Indeed, carefully analyzing the performances of the quadtree dissection and the number of required portals to achieve a $(1+\varepsilon)$-factor approximation has been a fruitful direction of research to design approximation schemes in Euclidean space. The initial work of Arora [3, 4] showed approximation schemes for the Traveling Salesperson Problem (TSP) and other routing problems, first in quasi-polynomial time (for $d > 2$) and then with running time $n\log(n)^{O(1/\varepsilon)^{d-1}}$. For TSP, the running time was subsequently significantly improved: first by Rao and Smith [45] who achieved $(1/\varepsilon)^{1/\varepsilon^{d-1}}n\log n$, then by Bartal and Gottlieb [7] who brought it down to $2^{1/\varepsilon^{O(d)}}n$, linear in n, but suboptimal in d. Finally, [34] settled the exact dependency on d, by showing an algorithm running essentially in time $2^{O\left(1/\varepsilon^{d-1}\right)}n$, and a lower bound of $2^{\Omega\left((1/\varepsilon)^{d-1}\right)}\mathrm{poly}(n)$ under the Gap-ETH hypothesis [25, 42].

In light of this literature, our contribution is therefore an improved analysis of the quadtree dissection for clustering problems, that allows us to (almost) match the number of portals required for the well-studied TSP. This requires an analysis and techniques distinct from those for TSP, in particular because they fail if distances are squared – as what happens in the k-means problem. We note that, due to its simplicity, applications of the quadtree extend far beyond approximation algorithms in low-dimensional space. For clustering, quadtrees are used for instance in the streaming scenario, when memory has to be kept low [8], but also to design linear time algorithms in high-dimensional spaces [21] or to construct differentially-private algorithms [12]. Therefore, better understanding the quadtree's performance, even in the simplest scenario, is likely to have a far-reaching impact.

We complement Theorem 2 with a conditional lower bound, which shows the near-optimality of our algorithm under the Gap-ETH hypothesis:

▶ **Theorem 3.** *Assuming the Gap-ETH hypothesis, for every integer $d \geq 2$, there exists a constant $c > 0$ such that no approximation scheme can, given an instance of discrete k-means (or discrete k-median) with N points in Euclidean space $\mathbb{R}^d$ and a parameter $\varepsilon > 0$, compute a $(1+\varepsilon)$-factor approximation in time $2^{c(1/\varepsilon)^{d-1}}\mathrm{poly}(N)$.*

This lower bound is obtained by amalgamating the technical ideas of [37, 11, 17] and the framework of de Berg, Bodlaender, Kisfaludi-Bak, Marx and van der Zanden [24]. Ideas to reduce the Vertex Cover problem to clustering problems were elaborated in [37, 11, 17], and [24] built a framework to prove fine-grained lower bounds for a host of geometric problems in low dimensions. We put them together to obtain Theorem 3. The proof is deferred to the full version.

Extensions. The framework developed in [13] applies to many variants of k-means (and k-median), namely prize-collecting k-means, k-means with outliers and Facility location. As our structure theorem is a direct improvement of theirs, we also improve the running time

for those problems. We give a sketch of the arguments in the full version. Our analysis also improves the result for doubling metric (a generalization of Euclidean space), however attaining a non-tight complexity $2^{\widetilde{O}(1/\varepsilon^d)}$. We therefore focused this paper on the Euclidean setting, for which our analysis is almost tight.

1.2 Our Techniques

For the ease of presentation, we detail below our upper bound for the k-means problem in the Euclidean plane (i.e., with $d = 2$). Our input is a set of points $P \subset \mathbb{R}^2$, a set of candidate centers $\mathcal{C} \subset \mathbb{R}^2$, and the goal is to compute a set of centers $S \in \mathcal{C}^k$ ($\mathcal{C}^k$ is the k-wise partition product of $\mathcal{C}$) that minimizes the sum of squared distances from points of P to their closest center in S.

Our starting point is the quadtree dissection equipped with portals. We briefly recall in the next few paragraphs its construction and main property.

First, find an axis-aligned rectangle that contains all input points. Then, randomly split the rectangle into 4 axis-aligned rectangles of side length reduced by a factor of approximately 2. Continue this splitting recursively until each rectangle contains a single point: this defines a recursive decomposition of the input. We say that two points are *cut at level i* when the smallest rectangle containing both points is at the i^{th} level of the decomposition. Assuming that the leaves of the decomposition have diameter 1, the rectangles at the i^{th} level have diameter roughly 2^i.

Now, a set of *portals* is placed regularly along the boundary of each rectangle. Instead of connecting points to centers via straight lines, we consider only *portal-respecting* paths, defined as follows. To connect two points p and q cut at level i, a portal-respecting path consists of a sequence of segments connecting $p, p_1, ..., p_i, q_i, ..., q_1, q$, where p_j (resp. q_j) is a portal of the rectangle at level j containing p (resp. q). Considering only those paths enforces some detours, but a simple dynamic program can compute the best portal-respecting solution, whose complexity naturally depends (exponentially) on the number of portals. Furthermore, if $1/\rho$ many portals are placed along each boundary, it can be computed that a portal-respecting path between two points p, q cut at level i has length (essentially) $\|p - q\| + \rho 2^i$. Indeed, since the boundaries of the rectangles are geometrically decreasing, the detour to go through a portal at level i dominates all others; and since the rectangles at this level have diameter 2^i, by regularly placing $1/\rho$ many portals, we can ensure that there is a portal at distance at most $\rho 2^i$ from the point where the straight line path crosses the boundary. Therefore, the goal of the analysis is to show that, for a small number of portals, a portal-respecting solution with almost optimal cost exists.

The standard argument goes as follows. The main property of the quadtree decomposition is that, for any pair of points $p, q \in \mathbb{R}^2$, the probability that they are cut at level i is $\|p-q\|/2^i$. By using $1/\rho$ portals, the detour to connect p and q via portals in case they are cut at level i is $\rho 2^i$. Therefore, the expected detour at level i is $\|p - q\|/2^i \cdot \rho 2^i = \rho\|p - q\|$. Since there are $\log n$ levels in the decomposition, the expected detour is $\rho\|p - q\| \cdot \log n$, and one has to take $\rho = \varepsilon/\log n$ to get a tiny expected detour.

However, having $\log n/\varepsilon$ many portals does not yield an efficient algorithm. More importantly, this analysis does not extend to k-means problem: although the expected distance between p and q is small, the expected *squared* distance is not! Indeed, the expected squared detour at level i is $\|p - q\|/2^i \cdot (\rho 2^i)^2 = \rho^2\|p - q\|2^i$, and the term 2^i cannot be easily controlled. To cope with this, [13] introduced a careful preprocessing of the input, to ensure that no point is cut at a level higher than $\|p - q\|/\varepsilon$. In that case, taking $1/\varepsilon^2$ portals is enough to ensure a small detour – and also small squared detour, as this is not an average-case argument (unlike the previously seen argument). This preprocessing is

essentially the following: start from a constant-factor approximate solution $\mathcal{A}$ (which is a set of k centers), and for each point p, let $\mathcal{A}_p$ be its distance to the closest center of $\mathcal{A}$. If the ball $B(p, \mathcal{A}_p/\varepsilon)$ is cut by the decomposition at a level higher than $\log(\mathcal{A}_p/\varepsilon^2)$, then replace p by a copy of its closest center in $\mathcal{A}$. They showed that each point is replaced with probability at most ε, therefore yielding an expected cost increase of $\varepsilon\text{cost}(P, \mathcal{A}) = O(\varepsilon)\text{OPT}$. Furthermore, [13] showed that placing $1/\varepsilon^3$ many portals ensured that, for any center of $\mathcal{A}$, either it can be connected to the optimal solution through portals with a small detour, or one can assume that the center is part of the optimal solution. Therefore, after preprocessing, each client can be connected through portals to the optimal solution, and dynamic programming allows to compute a near-optimal solution.

However, this analysis may be seen as "worst-case", in the sense that it treats all the clients for which the ball is cut at a level below $\log(\mathcal{A}_p/\varepsilon^2)$ in the same way. Our contribution is to mix the average-case analysis with the techniques of [13] in order to reduce the number of necessary portals. We carefully define a budget for each point, according to the level where the point is cut from the approximate solution $\mathcal{A}$, as in [13], but now also according to the level where it is cut from the optimal solution. We first show that, with constant probability, this budget is very cheap, namely an ε-fraction of the optimal cost. We then show that this budget is enough to pay for the preprocessing of the input, and for the detour incurred by making a solution portal-respecting. The algorithm is the same as the one of [13]: our contribution lies in a much tighter analysis.

This analysis differs from [13] as the detour they tolerated was independent of the optimal solution: using this additional information allows us to be more precise, and reduce the number of required portals to $(\log(1/\varepsilon)/\varepsilon)^{d-1}$, instead of $1/\varepsilon^{O(d)}$. Although the global picture remains similar, finding the right balance between a budget (1) small enough to not blow up the cost, and (2) that is still enough to pay for making the optimal solution portal-respecting is a non-trivial extension of their proof. Indeed, it requires new insights on the way to connect a point p that is *badly cut*, namely cut from its optimal center s at a level higher than $\log(\|p-s\|/\varepsilon)$. For this, we show that our budget allows to connect the point in a portal-respecting way to the center $\widetilde{s}$ closest to $\mathcal{A}(p)$, where $\mathcal{A}(p)$ is the closest center to p in the constant-factor approximate solution $\mathcal{A}$.

1.3 Further Related Work

We covered previously the hardness results for k-means and k-medians; in terms of upper-bound, for general metrics a recent breakthrough from [15] achieves a $(2+\varepsilon)$-approximation for k-median, and a 5.6-approximation for k-means [9]. Those two works improve a long line of works, based on primal-dual techniques and the so-called Lagrangian-multiplier preserving algorithms for Facility Location, see e.g. [33, 32, 38].

To improve those approximation ratio, one can resort to algorithms parameterized by the number of clusters, k: one can get FPT algorithm with approximation matching the lower bound of [32], namely $1+2/e$ for k-median and $1+8/e$ for k-means [16].

Other techniques, specific to Euclidean Space, have been developed in the past decades. Dimension reduction [41] and coreset [22] allow to reduce the dimension of the input to $O(\log k/\varepsilon^{-2})$ and the number of distinct points to $\widetilde{O}(k\varepsilon^{-O(1)})$, hence leading to simple FPT algorithms based on naive enumeration. Recent papers improved this simple approach, reducing the dependency in k (in the exponent) [1].

We covered in the introduction most of the literature related on low-dimensional clustering – which is the study of the problems parameterized by d. To compute a $(1+\varepsilon)$-approximation in time polynomial in n and k, any algorithm must have a running time at least doubly exponential in d, as the problem is APX-hard in dimension $\Omega(\log n)$. The best of these algorithms is from [13], with a near-linear running time of $f(\varepsilon, d) \cdot n \operatorname{polylog} n$.

Finally, tractability of clustering has also been studied with different parameters, e.g., the cost by Fomin, Golovach and Simonov [26], who showed a $D^D\text{poly}(nd)$ exact algorithm for k-median, where D is the optimal cost.

We present in the full version how to adapt the argument for k-median. The main body contains the proof of the structure theorem that allows to reduce the number of portals required by the quadtree. We extend the algorithmic results to other objectives in the full version.

2 Preliminaries

2.1 Definitions

We consider the Euclidean space $\mathbb{R}^d$, with the ℓ_2 metric: $\text{dist}(p,q) := \sqrt{\sum_{i=1}^d (p_i - q_i)^2}$. For any point p and $r \geq 0$, the ball centered at p of radius r is $B(p,r) := \{x \in \mathbb{R}^d : \text{dist}(p,x) \leq r\}$. The (discrete) k-median and k-means problems are defined as follows: we are given a set of *clients* P and of *candidate centers* $\mathcal{C}$. A *solution* is any subset of $\mathcal{C}$ with size k. The goal is to compute the solution $\mathcal{S}$ that minimizes $\text{cost}(P,\mathcal{S}) := \sum_{p\in P} \text{dist}(p,\mathcal{S})^z$, with $z = 1$ for k-median and $z = 2$ for k-means. We say a solution is an α-approximation if its cost is at most α times the minimal cost.

▶ Remark 4. We consider in this paper the *discrete* Euclidean clustering. In the continuous version, centers can be placed arbitrarily in the space. A classical result shows how to reduce continuous to discrete, by computing a set of $1/\varepsilon^d \cdot |P|$ candidate centers that contain a $(1+\varepsilon)$-approximation [43].

For some set of points $\mathcal{S}$, and any point $p \in \mathbb{R}^d$, we define $\mathcal{S}(p) = \arg\min_{s\in\mathcal{S}} \text{dist}(p,s)$ be the closest point in $\mathcal{S}$ to p ; and we let $\mathcal{S}_p := \text{dist}(p,\mathcal{S}(p))$ be the distance from p to its closest point in $\mathcal{S}$.

We assume that the minimal distance between two points is 1, and denote $\text{diam}(P)$ the maximum distance between two points of P. We also assume that $P \subset B(0,\text{diam}(P))$. Using standard arguments for clustering problems, we can assume $\text{diam}(P) = \text{poly}(n)$ [14].

A hierarchical decomposition $\mathcal{D}$ is a sequence $\mathcal{B}_1, \mathcal{B}_2 \ldots$ such that for any i, $\mathcal{B}_i$ is a partition of P, and $\mathcal{B}_i$ is a refinement of $\mathcal{B}_{i+1}$: namely, every part of $\mathcal{B}_i$ is fully contained in a part of $\mathcal{B}_{i+1}$ (note that in this definition, parts are increasing with i).

2.2 The Hierarchical Decomposition and Its Properties

▶ **Lemma 5.** *For any set of points $P \subset \mathbb{R}^d$ and any $\rho > 0$, there is a randomized hierarchical decomposition $\mathcal{D}$ such that the diameter of a part $B \in \mathcal{B}_i$ is at most 2^{i+1}, $|\mathcal{D}| \leq \lceil \log(\text{diam}(P)) \rceil$, each part $B \in \mathcal{B}_i$ is refined in at most $2^{O(d)}$ parts at level $i-1$, and:*

1. *for any $p \in \mathbb{R}^d$, radius r, and level i, we have*

$$\Pr[\mathcal{D} \text{ cuts } B(p,r) \text{ at a level } i] \leq dr/2^i.$$

2. *for every part $B \in \mathcal{B}_i$ of the decomposition, there is a set of at most $1/\rho^{d-1}$ many* portals *such that for every points $p \in B, q \notin B$ there is a portal $x \in \mathcal{P}_B$ close to the line connecting p to q, i.e., $\|p-x\| + \|x-q\| \leq \|p-q\| + \rho 2^{i+1}$. Furthermore, any portal of level $i+1$ that lies in B is also a portal of B.*

This decomposition can be found in time $(1/\rho)^{O(d)} n \log(\text{diam}(P))$.

In Euclidean space, this decomposition is precisely the quadtree dissection. The previous statement is an adaptation of Lemma 2.3 in [13] to Euclidean space. The improved Property 1 is shown, e.g., in Lemma 11.3 of [30]. Since $\text{diam}(P) = \text{poly}(n)$, the decomposition can be computed in near-linear time.

We denote $\tau(\varepsilon, d) = \log(d) + \log(1/\varepsilon)$, a parameter often used throughout this paper.

A *portal-respecting* path between two points p and q cut at level i consists of a sequence of segments connecting $p, p_1, ..., p_i, q_i, ..., q_1, q$, where p_j (resp. q_j) is a portal of the rectangle at level j containing p (resp. q). By extension, a solution to k-means is portal-respecting if all paths connecting clients to center are portal-respecting. Recall that $\mathcal{A}_p$ is the distance from p to the closest point in a solution $\mathcal{A}$. An important definition we will use is that of badly cut, slightly modified from [13]:

▶ **Definition 6.** *Let $P \subset \mathbb{R}^d$, $\mathcal{D}$ be a hierarchical decomposition of P, and $\varepsilon > 0$.*

For any ball $B(x, r)$, we define $\mathrm{lvl}(x, r)$ *the level at which the ball $B(x, r)$ is cut by $\mathcal{D}$.*

We say a ball $B(x, r)$ is badly cut w.r.t. $\mathcal{D}$ *if* $\mathrm{lvl}(x, r) \geq \log(3r) + \tau(\varepsilon, d)$. *We say a point p is* badly cut w.r.t. $\mathcal{D}$ and $\mathcal{A}$ *if $B(p, 3\mathcal{A}_p)$ is badly cut w.r.t. $\mathcal{D}$ (namely,* $\mathrm{lvl}(p, 3\mathcal{A}_p) \geq \log(3\mathcal{A}_p) + \tau(\varepsilon, d)$*).*

In the following, we will consider badly cut points w.r.t. two different sets: $\mathcal{A}$ will be a constant factor approximation, known to the algorithm, and $\mathcal{S}$ a slight modification of the optimal solution, unknown to the algorithm, that we will specify later; those two solutions being independent of the decomposition $\mathcal{D}$. For input points in P, we will examine whether they are badly cut w.r.t. $\mathcal{D}$ and $\mathcal{A}$. For centers of $\mathcal{A}$, it will be w.r.t. $\mathcal{D}$ and $\mathcal{S}$. Furthermore, $\mathcal{D}$ will always be the same, hence we say for simplicity badly cut w.r.t. $\mathcal{A}$ (or w.r.t. $\mathcal{S}$).

The following lemma bounds the probability of being badly cut, and is a direct consequence of Property 1.

▶ **Lemma 7.** *Let $P, S \subset \mathbb{R}^d$ and $\mathcal{D}$ a random hierarchical decomposition given by Lemma 5, with $\rho = O(\varepsilon/\log(1/\varepsilon))$. For any point $p \in \mathbb{R}^d$, the probability that p is badly cut w.r.t. S is at most ε.*

Given a random decomposition, each point is assigned a *budget*. In the following definition, one can think of $\mathcal{A}$ as an $O(1)$-approximation known by the algorithm, and $\mathcal{S}$ as the optimal solution. We show in Lemma 12 that this budget is small, and that one can compute a solution with cost at most the optimal cost plus the budget.

▶ **Definition 8.** *Let $P \subset \mathbb{R}^d$, $\mathcal{D}$ be a hierarchical decomposition of P, $\varepsilon > 0$, $\mathcal{A}$ and $\mathcal{S}$ be two solutions to k-means on P.*

For a ball $B(x, r)$, its detour *with respect to B and $\mathcal{D}$ is* $\det_{\mathcal{D}}(x, r) = \varepsilon 2^{\mathrm{lvl}(x,r)} r + \varepsilon^2 2^{2\,\mathrm{lvl}(x,r)}$.

A point $p \in P$ has budget *with respect to $\mathcal{A}$ and $\mathcal{S}$ $b(p, \varepsilon) = b_1(p, \varepsilon) + b_2(p, \varepsilon) + b_3(p, \varepsilon)$, with*

$$b_1(p,\varepsilon) = \begin{cases} \det_{\mathcal{D}}(p, 3\mathcal{A}_p) \text{ if } \mathrm{lvl}(p, 3\mathcal{A}_p) \leq \log(3\mathcal{A}_p) + \tau(\varepsilon, d) \\ 0 \text{ otherwise} \end{cases}$$

$$b_2(p,\varepsilon) = \begin{cases} \det_{\mathcal{D}}(p, 3(\mathcal{A}_p + \mathcal{S}_p)) \text{ if } \mathrm{lvl}(p, 3(\mathcal{A}_p + \mathcal{S}_p)) \leq \log(3(\mathcal{A}_p + \mathcal{S}_p)) + \tau(\varepsilon, d) \\ 36d\mathcal{A}_p^2 + 16d\mathcal{S}_p^2 \text{ otherwise} \end{cases}$$

$$b_3(p,\varepsilon) = \begin{cases} \det_{\mathcal{D}}(\mathcal{A}(p), 3\mathcal{S}_{\mathcal{A}(p)}) \text{ if } \mathrm{lvl}(\mathcal{A}(p), 3\mathcal{S}_{\mathcal{A}(p)}) \leq \log(3\mathcal{S}_{\mathcal{A}(p)}) + \tau(\varepsilon, d) \\ 0 \text{ otherwise} \end{cases}$$

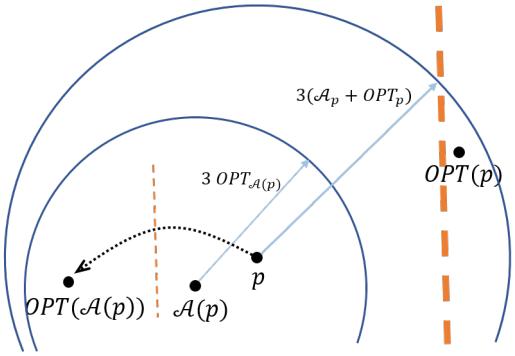


Figure 1 Illustration of one case of the distinction. $B(p, 3(\mathcal{A}_p + \mathrm{OPT}_p))$ is badly cut by the thick orange-dashed line, so p cannot be connected via portals to $\mathrm{OPT}(p)$. However, p and $\mathcal{A}(p)$ are not badly cut, so p can be connected to $\mathrm{OPT}(\mathcal{A}(p))$ instead, making a detour through the thin orange-dashed line. The cost of this reassignment is charged to $b_2(p)$.

3 Structure Theorem and the Approximation Algorithm

3.1 Roadmap

Now that all definitions are in place, we give a more precise high-level overview of the proof of Theorem 2. Our goal is to show that the decomposition of Lemma 5 with $\rho = \varepsilon/\log(1/\varepsilon)$ is precise enough for a dynamic program to compute a portal-respecting solution that is a $(1+\varepsilon)$-approximation.

To do this, we show that one can find a solution $\mathcal{S}^*$ such that: (1) it is a $(1+\varepsilon)$-approximation, and (2) the detour from each point to $\mathcal{S}^*$ is at most the budget of the point. To conclude, it is enough to bound the budget, which is a mere application of Lemma 5 and first-moment analysis. To find such a solution $\mathcal{S}^*$, we start from the optimal solution (which respects (1) but not (2)), and transform it such that each client can be connected through the portals.

First, consider the easy case where a client p is not badly cut, and $B(p, 3(\mathcal{A}_p + \mathrm{OPT}_p))$ is not badly cut. Then, the detour from p to OPT_p can be charged to $b_2(p, \varepsilon)$, as the closest optimal center to p lies in the latter ball. When this ball is badly cut, instead of serving p by its closest center $\mathrm{OPT}(p)$, we serve it by $\mathrm{OPT}(\mathcal{A}(p))$. We show that this can always be charged to the budget of p. Indeed, in the sub-case where $\mathcal{A}(p)$ itself is not badly cut, we can show that the detour between p and $\mathrm{OPT}(\mathcal{A}(p))$ is affordable compared to the distance between them; and this distance can be charged to the budget $b_2(p, \varepsilon)$. This sub-case is illustrated in Figure 1. However, in the sub-case where $\mathcal{A}(p)$ is badly cut as well, we have no choice but to add the center to OPT. This results in a solution with slightly too many centers – in expectation $(1+\varepsilon)k$, since each center is badly cut with probability ε – and we will need to remove some centers of the optimal solution without increasing the cost too much. This step is already presented in [13], and we recall it in Section 3.2.

This essentially concludes the case where p is not badly cut. When it is badly cut, the decomposition $\mathcal{D}$ approximates its distances poorly; however, this happens with tiny probability, and we can afford to move p to $\mathcal{A}(p)$ (charging this movement to $b_1(p, \varepsilon)$). A case analysis, similar to the previous one, then shows that the detour between $\mathcal{A}(p)$ to OPT can be charged to $b_2(p, \varepsilon)$ and $b_3(p, \varepsilon)$.

In the remainder of this section, we show how to remove some centers from the optimal solution (to then add the badly cut centers of $\mathcal{A}$), then prove the structure theorem that shows the existence of a solution satisfying (1) and (2). We explain in Section 3.4 the dynamic program that relies on this structure theorem and enables the computation of a good approximation, to conclude the proof of Theorem 2.

3.2 Preparing the Instance

We consider the mapping of the centers of OPT to $\mathcal{A}$ defined as follows: for any $f \in \text{OPT}$, let $\mathcal{A}(f)$ denote the center of $\mathcal{A}$ that is the closest to f. Recall that for a client c, $\mathcal{A}(c)$ is the center serving c in $\mathcal{A}$.

For any center ℓ of $\mathcal{A}$, define $\psi(\ell)$ to be the set of centers of OPT that are mapped to ℓ, namely, $\psi(\ell) = \{f \in \text{OPT} \mid \mathcal{A}(f) = \ell\}$. Define $\mathcal{A}^1$ to be the set of centers ℓ of $\mathcal{A}$ for which there exists a unique $f \in \text{OPT}$ such that $\mathcal{A}(f) = \ell$, namely $\mathcal{A}^1 = \{\ell \in \mathcal{A} \mid |\psi(\ell)| = 1\}$. Let $\mathcal{A}^0 = \{\ell \in \mathcal{A} \mid |\psi(\ell)| = 0\}$, and $\mathcal{A}^{\geq 2} = \mathcal{A} \setminus (\mathcal{A}^1 \cup \mathcal{A}^0)$. Similarly, define $\text{OPT}^1 = \{f \in \text{OPT} \mid \mathcal{A}(f) \in \mathcal{A}^1\}$ and $\text{OPT}^{\geq 2} = \{f \in \text{OPT} \mid \mathcal{A}(f) \in \mathcal{A}^{\geq 2}\}$. Note that $|\text{OPT}^{\geq 2}| = |\mathcal{A}^0| + |\mathcal{A}^{\geq 2}|$, since $|\text{OPT}^1| = |\mathcal{A}^1|$ and, w.l.o.g., $|\text{OPT}| = |\mathcal{A}| = k$.

As shown in [13], one can remove some of the centers from $\text{OPT}^{\geq 2}$ to obtain the following result:

▶ **Lemma 9** (See Step 1. and Lemma 4.1 in [13]). *There exists a set $\mathcal{H} \subseteq OPT^{\geq 2}$ such that:*

- $|\mathcal{H}| = \lfloor \varepsilon \cdot |OPT^{\geq 2}|/2 \rfloor$,
- *for any $\ell \in \mathcal{A}$, the center in $\psi(\ell)$ closest to ℓ is not in $\mathcal{H}$,*
- *let $OPT' = OPT \setminus \mathcal{H}$: OPT' has cost at most $(1 + O(\varepsilon))\text{cost}(OPT) + O(\varepsilon)\text{cost}(\mathcal{A})$.*

Furthermore, we transform the input set P to give it more structure, as follows. Given a randomized hierarchical decomposition $\mathcal{D}$ and a solution $\mathcal{A}$, we define the set $\widetilde{P}$ as follows. For every point p,

$$\widetilde{p} = \begin{cases} \mathcal{A}(p) \text{ if } p \text{ is badly cut w.r.t. } \mathcal{D} \text{ and } \mathcal{A} \\ p \text{ otherwise.} \end{cases}$$

We note that $\widetilde{P} := \{\widetilde{p} : p \in P\}$ is a random variable that depends on the randomness of $\mathcal{D}$.

3.3 Construction of a Structured Solution

Using Lemma 9, we can now show the existence of a near-optimal portal-respecting solution. Let $\mathcal{A}$ be a fixed solution, and OPT' the solution from Lemma 9 for P. Let $\mathcal{D}$ be a randomized hierarchical decomposition, and $B_{\mathcal{D}}$ be the set of centers of $\mathcal{A}$ that are badly cut w.r.t. OPT', i.e., $c \in B_{\mathcal{D}}$ when the ball $B(c, 3\text{OPT}'_c)$ is cut at some level greater than $\log(3\text{OPT}'_c) + \tau(\varepsilon, d)$.

▶ **Definition 10.** *For $\varepsilon > 0$, we say that $\mathcal{D}$ has* ε-small distortion *if*

1. *the total budget w.r.t. $\mathcal{A}$ and OPT' is bounded: $\sum_{p \in P} b(p, \varepsilon) \leq \varepsilon(\text{cost}(P, OPT) + \text{cost}(P, \mathcal{A}))$.*
2. *there exists a solution $\mathcal{S}^*$ that contains $B_{\mathcal{D}}$ with $\sum_{p \in P} (\text{dist}(p, \widetilde{p}) + \text{dist}(\widetilde{p}, \mathcal{S}^*))^2 \leq (1 + \varepsilon)\text{cost}(P, OPT) + \varepsilon\text{cost}(P, \mathcal{A})$,*
3. *for each point $p \in P$, its detour to $\mathcal{S}^*$ is at most $b(p, \varepsilon)$, namely there is a center $s \in \mathcal{S}^*$ cut from $\widetilde{p}$ at level i such that $(\text{dist}(p, \widetilde{p}) + \text{dist}(\widetilde{p}, s) + \varepsilon 2^i)^2 \leq (\text{dist}(p, \widetilde{p}) + \text{dist}(\widetilde{p}, \mathcal{S}^*))^2 + b(p, \varepsilon)$.*

As explained in the introduction, we will use those properties as follows: a standard dynamic program will compute the best *portal-respecting* solution that minimizes $\widetilde{\text{cost}}(P,\mathcal{S}) := \sum_p (\text{dist}(p,\widetilde{p}) + \text{dist}(\widetilde{p},\mathcal{S}))^2$. Making the solution portal-respecting incurs an error $\varepsilon 2^i$ for two points cut at level i: Property 3 therefore bounds this error by $b(p,\varepsilon)$. Property 1 of Definition 10 ensures that the total budget is bounded, and Property 2 relates $\widetilde{\text{cost}}$ to cost.

Our main structural theorem for k-means in low dimension is the following:

▶ **Theorem 11** (Structure Theorem). *For any $\varepsilon > 0$, $\mathcal{D}$ has ε-small-distortion with probability at least $2/3$.*

The rest of the paper is dedicated to the proof of this theorem.

3.3.1 Bounding the Budget

▶ **Lemma 12.** *Fix two solutions $\mathcal{A}$ and $\mathcal{S}$, and let $b(p,\varepsilon)$ be the budget of p w.r.t. solutions $\mathcal{A}$ and $\mathcal{S}$. With probability $2/3$ (over the randomness of $\mathcal{D}$), $\sum_p b(p,\varepsilon) = O(d^2\log(1/\varepsilon)\cdot\varepsilon)(\text{cost}(P,\mathcal{A}) + \text{cost}(P,\mathcal{S}))$*

Proof. We bound the expectation of the budget and then use Markov's inequality. Recall that any ball $B(x,r)$ is cut at level i with probability $dr/2^i$, and is badly cut with probability ε. Furthermore, when $B(x,r)$ is cut at level i, then $\det_{\mathcal{D}}(x,r) = \varepsilon r 2^i + \varepsilon^2 2^{2i}$.

To simplify the analysis slightly, we first analyze the expectation of the variable

$$\beta(x,r) := \begin{cases} \det_{\mathcal{D}}(x,r) \text{ when } \text{lvl}(x,r) \le \log(r) + \tau(\varepsilon,d) \\ 0 \text{ otherwise.} \end{cases}$$

To do so, we distinguish between the cases where $\text{lvl}(x,r)$ is at most $\log(r)$, between $\log(r)$ and $\log(r)+\tau(\varepsilon,d)$, and at least $\log(r)+\tau(\varepsilon,d)$. This gives the following upper bound:

$$\begin{aligned}
\mathbb{E}[\beta(x,r))] &\le \sum_{i\le\log(r)} \det_{\mathcal{D}}(x,r) + \sum_{i=\log(r)}^{\log(r)+\tau(\varepsilon,d)} \det_{\mathcal{D}}(x,r)\cdot\Pr[\mathcal{D}\text{ cuts } B(p,r) \text{ at a level } i] \\
&\le \sum_{i\le\log(r)} \left(\varepsilon 2^i r + \varepsilon^2 2^{2i}\right) + \sum_{i=\log(r)}^{\log(r)+\tau(\varepsilon,d)} \left(\varepsilon 2^i r + \varepsilon^2 2^{2i}\right)\cdot\frac{dr}{2^i} \\
&\le 3\varepsilon r^2 + \sum_{i=\log(r)}^{\log(r)+\tau(\varepsilon,d)} \left(\varepsilon d r^2 + \varepsilon^2 2^i\cdot dr\right) \\
&\le 3\varepsilon r^2 + \tau(\varepsilon,d)\cdot\varepsilon d r^2 + \varepsilon^2\cdot dr\cdot 2^{\log(r)+\tau(\varepsilon,d)} \\
&= O(d^2\log(1/\varepsilon))\varepsilon r^2,
\end{aligned}$$

where the last line uses $2^{\tau(\varepsilon,d)} = 2^{\log(d)+\log(1/\varepsilon)} = O(d/\varepsilon)$.

Since $b_1(p,\varepsilon) = \beta(p,3\mathcal{A}_p)$ and $b_3(p,\varepsilon) = \beta(\mathcal{A}(p),3\mathcal{S}_{\mathcal{A}(p)})$, we directly get $\mathbb{E}[b_1(p,\varepsilon) + b_3(p,\varepsilon)] \le O(d^2\log(1/\varepsilon)\varepsilon)(\mathcal{A}_p^2 + \mathcal{S}_p^2)$. Bounding $b_2(p,\varepsilon)$ is only slightly more complicated: $b_2(p,\varepsilon) = \beta(p,\mathcal{A}_p+\mathcal{S}_p) + X(p,\varepsilon)$, where $X(p,\varepsilon) := 36d\mathcal{A}_p^2 + 16d\mathcal{S}_p^2$ when $\text{lvl}(p,3(\mathcal{A}_p+\mathcal{S}_p)) > \log(3(\mathcal{A}_p+\mathcal{S}_p)) + \tau(\varepsilon,d)$. Therefore, it is enough to compute $\mathbb{E}[X(p,\varepsilon)]$.

Properties of the decomposition in Lemma 5 ensure that $\text{lvl}(p,3(\mathcal{A}_p+\mathcal{S}_p)) > \log(3(\mathcal{A}_p+\mathcal{S}_p)) + \tau(\varepsilon,d)$ with probability at most ε, and therefore $\mathbb{E}[X(p,\varepsilon)] \le \varepsilon\cdot(36d\mathcal{A}_p^2 + 16d\mathcal{S}_p^2)) = O(\varepsilon d(\mathcal{A}_p^2+\mathcal{S}_p^2))$.

Therefore, $\mathbb{E}[b_2(p,\varepsilon)] = \mathbb{E}[\beta(p,3(\mathcal{A}_p+\mathcal{S}_p)] + \mathbb{E}[X(p,\varepsilon)] \le O(d^2\log(1/\varepsilon)\cdot\varepsilon)(\mathcal{A}_p^2+\mathcal{S}_p^2)$.

Summing these inequalities for b_1, b_2, b_3, and for all $p\in P$ we get $\mathbb{E}[\sum_{p\in P} b(p,\varepsilon)] \le O(d^2\log(1/\varepsilon)\cdot\varepsilon)(\text{cost}(P,\mathcal{A}) + \text{cost}(P,\mathcal{S}))$. Markov's inequality concludes the lemma. ◀

3.3.2 Construction of $\mathcal{S}^*$

To construct $\mathcal{S}^*$, we start from the solution OPT' obtained from Lemma 9 and apply the two following steps.

For any $\ell \in \mathcal{A}$, define f_ℓ to be its closest center in $\psi(\ell)$, breaking ties arbitrarily. We start with $\mathcal{S}^* = \mathrm{OPT}'$ obtained from Lemma 9. Note that for every $\ell \in \mathcal{A}^1 \cup \mathcal{A}^{\geq 2}$, the second property of Lemma 9 ensures that f_ℓ is a center of OPT'. $\mathcal{S}^*$ is then transformed as follows:

- **Step 1.** For each center $\ell \in B_{\mathcal{D}} \setminus \mathcal{A}^0$ (i.e., ℓ is badly cut w.r.t. $\mathcal{D}$ and OPT', and $\psi(\ell) \neq \emptyset$), replace f_ℓ by ℓ in $\mathcal{S}^*$.
- **Step 2.** Add all centers of $\mathcal{A}^0$ badly cut w.r.t. $\mathcal{D}$ and OPT' to $\mathcal{S}^*$.

The two key properties of that procedure are the following:

▶ **Fact 13** (Claim 4.3 and 4.4 in [13]). *With probability* $4/5$, $|\mathcal{S}^*| \leq k$, *so* $\mathcal{S}^*$ *is a valid solution, and* $\mathrm{cost}(P, \mathcal{S}^*) \leq (1+\varepsilon)\mathrm{cost}(P, OPT) + \varepsilon\mathrm{cost}(P, \mathcal{A})$. *Furthermore, for any point* p, *it holds that* $\mathrm{dist}(p, \mathcal{S}^*) \leq 2\,\mathrm{dist}(p, OPT') + \mathrm{dist}(p, \mathcal{A})$.

In addition, the construction of $\mathcal{S}^*$ ensures the below whose proof is in the full version:

▶ **Fact 14.** *For any point* p,

$$\mathrm{dist}(p, \widetilde{p}) + \mathrm{dist}(\widetilde{p}, \mathcal{S}^*) \leq 3\mathcal{A}_p + 2OPT'_p$$
$$\mathrm{dist}(p, \mathcal{S}^*(\mathcal{A}(p))) \leq 3\mathcal{A}_p + 2OPT'_p.$$

Furthermore, for any center $\ell \in \mathcal{A}$, *it holds that* $\mathrm{dist}(\ell, \mathcal{S}^*) \leq 2\,\mathrm{dist}(\ell, OPT')$.

We can now use this solution $\mathcal{S}^*$ to show $\mathcal{D}$ has ε-small distortion.

3.3.3 Bounding the cost of $\mathcal{S}^*$

▶ **Lemma 15.** $\mathcal{S}^*$ *contains* $B_{\mathcal{D}}$, *and with probability at least* $4/5$ *it holds that*

$$\sum_{p\in P} \left(\mathrm{dist}(p, \widetilde{p}) + \mathrm{dist}(\widetilde{p}, \mathcal{S}^*)\right)^2 \leq (1+\varepsilon)\mathrm{cost}(P, OPT) + \varepsilon\mathrm{cost}(P, \mathcal{A})$$

Proof. By construction, $\mathcal{S}^*$ does contain $B_{\mathcal{D}}$. To simplify the equations, we write p b.c. when p is badly cut w.r.t. $\mathcal{A}$ (in which case $\widetilde{p} = \mathcal{A}(p)$). It holds that

$$\sum_{p\in P} \left(\mathrm{dist}(p, \widetilde{p}) + \mathrm{dist}(\widetilde{p}, \mathcal{S}^*)\right)^2 \leq \sum_{p\in P} \mathrm{dist}(p, \mathcal{S}^*)^2 + \sum_{p \text{ b.c.}} \left(\mathrm{dist}(p, \widetilde{p}) + \mathrm{dist}(\widetilde{p}, \mathcal{S}^*)\right)^2. \tag{1}$$

By Fact 13, the first term is at most $(1+\varepsilon)\mathrm{cost}(P, \mathrm{OPT}) + \varepsilon\mathrm{cost}(P, \mathcal{A})$: thus, we focus only on the second term. For this, we use Fact 14:

$$\begin{aligned}\sum_{p \text{ b.c.}} \left(\mathrm{dist}(p, \widetilde{p}) + \mathrm{dist}(\widetilde{p}, \mathcal{S}^*)\right)^2 &\leq \sum_{p \text{ b.c.}} \left(3\mathcal{A}_p + 2\mathrm{OPT}'_p\right)^2 \\ &\leq 18 \sum_{p \text{ b.c.}} \mathcal{A}_p^2 + \mathrm{OPT}'^2_p\end{aligned}$$

Each point p is badly cut with probability ε: therefore, in expectation, it holds that

$$\begin{aligned}\mathbb{E}[18 \sum_{p \text{ b.c.}} \mathcal{A}_p^2 + \mathrm{OPT}'^2_p] &\leq 18\varepsilon \sum_{p\in P} \mathcal{A}_p^2 + \mathrm{OPT}'^2_p \\ &\leq 18\varepsilon(\mathrm{cost}(P, \mathcal{A}) + \mathrm{cost}(P, \mathrm{OPT}'))\end{aligned}$$

Using Markov's inequality, it holds with probability 4/5 that

$$\sum_{p \text{ b.c.}} \left(\text{dist}(p,\widetilde{p}) + \text{dist}(\widetilde{p},\text{OPT}')\right)^2 \leq 18 \cdot 5 \cdot \varepsilon(\text{cost}(P,\mathcal{A}) + \text{cost}(P,\text{OPT}')).$$

Plugging this bound into Equation (1) and using Lemma 9 to bound the cost of OPT', we conclude:

$$\begin{aligned}&\sum_{p\in P} (\text{dist}(p,\widetilde{p}) + \text{dist}(\widetilde{p},\mathcal{S}^*))^2 \\ &\quad\leq (1+\varepsilon)\text{cost}(P,\text{OPT}) + \varepsilon\text{cost}(P,\mathcal{A}) + 18\cdot 5\cdot\varepsilon(\text{cost}(P,\mathcal{A}) + \text{cost}(P,\text{OPT}')) \\ &\quad= (1+O(\varepsilon))\text{cost}(P,\text{OPT}) + O(\varepsilon)\text{cost}(P,\mathcal{A}).\end{aligned}$$

◀

3.3.4 Bounding the detour

We now show that the detour incurred by connecting $\widetilde{P}$ to $\mathcal{S}^*$ through portals of the decomposition $\mathcal{D}$ is within the budget defined in Definition 8.

▶ **Lemma 16.** *Let $P \subset \mathbb{R}^d$, $\mathcal{A}$ be a solution for k-means on P and $\mathcal{D}$ be a randomized decomposition. Let $\mathcal{S}^*$ be the solution from Lemma 15.*

Then, for any $p \in P$, there is a center $s \in \mathcal{S}^*$ cut from $\widetilde{p}$ at some level i such that $(\text{dist}(p,\widetilde{p}) + \text{dist}(\widetilde{p},s) + \varepsilon 2^i)^2 \leq (\text{dist}(p,\widetilde{p}) + \text{dist}(\widetilde{p},\mathcal{S}^*))^2 + b(p,\varepsilon)$.

Proof. We recall that any point $x \in \mathbb{R}^d$ is badly cut w.r.t. $\mathcal{D}$ and a set S when the ball $B(x, 3\,\text{dist}(x,S))$ is cut at level higher than $\log(9\,\text{dist}(x,S)) + \tau(\varepsilon,d)$. We recall our construction of $\widetilde{P}$ and $\mathcal{S}^*$: When a point p is badly cut w.r.t $\mathcal{D}$ and $\mathcal{A}$, $\widetilde{p} = \mathcal{A}(p)$; when a center $\ell \in \mathcal{A}$ is badly cut w.r.t $\mathcal{D}$ and OPT', $\ell \in \mathcal{S}^*$.

To show the lemma, we make a case distinction, first according to whether p is badly cut or not w.r.t. $\mathcal{D}$ and $\mathcal{A}$. In the case where it is badly cut, it holds that $\widetilde{p} = \mathcal{A}(p)$. Then, we have the following cases.

1. If $\mathcal{A}(p)$ is badly cut w.r.t. $\mathcal{D}$ and OPT', then $\mathcal{A}(p) \in \mathcal{S}^*$: therefore, $\widetilde{p} = \mathcal{A}(p) = \mathcal{S}^*(p)$, and $\widetilde{p}$ and $\mathcal{S}^*(p)$ are not cut, e.g. $i = -\infty$. Therefore, $s = \mathcal{A}(p)$ satisfies the lemma, with $(\text{dist}(p,\widetilde{p}) + \text{dist}(\widetilde{p},s) + \varepsilon 2^i)^2 = \text{dist}(p,\widetilde{p})^2$.
2. $\mathcal{A}(p)$ is not badly cut w.r.t. $\mathcal{D}$ and OPT'. Let $s = \mathcal{S}^*(\mathcal{A}(p))$, and i be the level at which $\mathcal{A}(p)$ and s are cut. We write $(\text{dist}(p,\widetilde{p}) + \text{dist}(\widetilde{p},s) + \varepsilon 2^i)^2 = (\mathcal{A}_p + \text{dist}(\widetilde{p},\mathcal{S}^*))^2 + 2\varepsilon 2^i \cdot \mathcal{A}_p + 2\varepsilon 2^i \cdot \mathcal{S}^*_{\mathcal{A}(p)} + \varepsilon^2 2^{2i}$.

   By Fact 14, $s \in B(\mathcal{A}(p), 2\text{OPT}'_{\mathcal{A}(p)})$, and so $i \leq i' := \text{lvl}\left(\mathcal{A}(p), 3\text{OPT}'_{\mathcal{A}(p)}\right)$. Combined with the fact that $\mathcal{A}(p)$ is not badly cut, it therefore holds that $2\varepsilon 2^i \cdot \mathcal{S}^*_{\mathcal{A}(p)} + \varepsilon^2 2^{2i} \leq 2\varepsilon 2^{i'} \cdot 2\text{OPT}'_{\mathcal{A}(p)} + \varepsilon^2 2^{2i'} \leq b_3(p)$. Thus, it is enough to bound $2\varepsilon 2^i \cdot \mathcal{A}_p$. For this, we refine our case distinction, according to whether the ball $B(p, 3(\mathcal{A}_p + \text{OPT}'_p))$ is badly cut or not, and use the budget from $b_2(p)$.
   a. If the ball is not badly cut, then we remark that Fact 14 ensures that both s and $\mathcal{A}(p)$ are in $B(p, 3(\mathcal{A}_p + \text{OPT}'_p))$, and therefore $i \leq i'' := \text{lvl}(p, 3(\mathcal{A}_p + \text{OPT}'_p))$. Therefore, $2\varepsilon 2^i \cdot \mathcal{A}_p \leq \varepsilon 2^{i''} 3(\mathcal{A}_p + \text{OPT}'_p) \leq b_2(p)$.
   b. If this ball is badly cut, then we use $i \leq i' \leq \log(3\text{OPT}'_{\mathcal{A}(p)}) + \tau(\varepsilon,d)$ in addition to $\text{OPT}'_{\mathcal{A}(p)} \leq \mathcal{A}_p + \text{OPT}'_p$ as follows: $2\varepsilon 2^i \cdot \mathcal{A}_p \leq 6\varepsilon \cdot \text{OPT}'_{\mathcal{A}(p)} \cdot 2^{\tau(\varepsilon,d)}\mathcal{A}_p \leq 6d(\mathcal{A}_p + \text{OPT}'_p)\mathcal{A}_p \leq b_2(p)$.

   Combining these cases, we get that when $\mathcal{A}(p)$ is not badly cut, $(\text{dist}(p,\widetilde{p}) + \text{dist}(\widetilde{p},s) + \varepsilon 2^i)^2 \leq (\mathcal{A}_p + \text{dist}(\widetilde{p},\mathcal{S}^*))^2 + b_2(p) + b_3(p)$.

We now turn to the case where p is not badly cut. In that case, $\widetilde{p} = p$: we therefore only need to bound $\left(\text{dist}(p, \mathcal{S}^*) + \varepsilon 2^i\right)^2$ and the details of this are presented in the full version. ◀

3.4 The Algorithm: how to use the Structure Theorem

Suppose that $\mathcal{D}$ has ρ-small-distortion, for $\rho = O(\varepsilon/\log(1/\varepsilon))$ (where the O hides dependencies in d), and let $\mathcal{S}^*$ be the solution provided by Definition 10. As explained previously, our algorithm will be a standard dynamic program working as follows. The decomposition $\mathcal{D}$ will be augmented with a set of portals, as defined in Lemma 5, with $\rho = \varepsilon/\log(1/\varepsilon)$. The dynamic program (described later in greater detail) computes the best portal-respecting solution, namely, the solution where each path connecting a client to a center crosses boundaries of $\mathcal{D}$ only at portals. More precisely, it computes the portal-respecting solution with (almost) smallest $\widetilde{\text{cost}}(P, \mathcal{S}) := \sum_p (\text{dist}(p, \widetilde{p}) + \text{dist}(\widetilde{p}, \mathcal{S}))^2$

▶ **Theorem 17.** *There exists an algorithm running in time* $(2/\varepsilon)^{O\left(\log(1/\varepsilon)/\varepsilon\right)^{d-1}} \cdot n \log^3 n$ *that computes a portal-respecting solution* $\mathcal{S}$ *such that* $\widetilde{\text{cost}}(P, \mathcal{S}) \leq (1+\varepsilon) \min_{\mathcal{S}'} \widetilde{\text{cost}}(P, \mathcal{S}')$.

The preciseness and nestedness of portal sets ensure that, when two points x and y are cut at level i, then the portal-respecting path between x and y has length $\|x - y\| + \rho 2^{i+1}$. Furthermore, the algorithm from Theorem 17 computes a solution at least as good as $\mathcal{S}^*$: by Property 3 of Definition 10, the portal-respecting $\widetilde{\text{cost}}$ of $\mathcal{S}^*$ is at most $\sum_p (\text{dist}(p, \widetilde{p}) + \text{dist}(\widetilde{p}, \mathcal{S}^*))^2 + b(p, \rho)$. Property 1 ensures that the total budget is at most $\rho(\text{cost}(P, \text{OPT}) + \text{cost}(P, \mathcal{A})) = O(\rho)\text{cost}(P, \text{OPT})$, since $\mathcal{A}$ is an $O(1)$-approximation. Combined with Property 2 of Definition 10, this shows that the portal-respecting $\widetilde{\text{cost}}$ of the solution computed by the algorithm is at most $(1 + O(\varepsilon))\text{cost}(P, \text{OPT})$.

To wrap up, we note that the portal respecting $\widetilde{\text{cost}}$ is an upper bound on cost: going in straight lines instead of going through portals only improves the cost, and for any point p, $\text{dist}(p, \mathcal{S}) \leq \text{dist}(p, \widetilde{p}) + \text{dist}(\widetilde{p}, \mathcal{S})$. Therefore, the solution computed by the dynamic program is a $(1 + O(\varepsilon))$-approximation for k-means.

The algorithm used to show Theorem 17 is a quite standard dynamic program (DP), described in detail in [13] and is deferred to the full version.

References

1 Fateme Abbasi, Sandip Banerjee, Jaroslaw Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. Parameterized approximation schemes for clustering with general norm objectives. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1377–1399. IEEE, 2023. doi:10.1109/FOCS57990.2023.00085.

2 Enver Aman, Karthik C. S., and Sharath Punna. On connections between k-coloring and Euclidean k-means. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 9:1–9:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.9.

3 Sanjeev Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 2–11. IEEE Computer Society, 1996. doi:10.1109/SFCS.1996.548458.

4 Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

5 Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for Euclidean k-medians and related problems. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 106–113. ACM, 1998. doi:10.1145/276698.276718.

6 Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of Euclidean k-means. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, volume 34 of *LIPIcs*, pages 754–767. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.SOCG.2015.754.

7 Yair Bartal and Lee-Ad Gottlieb. A linear time approximation scheme for Euclidean TSP. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 698–706. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.80.

8 Vladimir Braverman, Gereon Frahling, Harry Lang, Christian Sohler, and Lin F. Yang. Clustering high dimensional dynamic data streams. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 576–585. PMLR, 2017. URL: http://proceedings.mlr.press/v70/braverman17a.html.

9 Moses Charikar, Vincent Cohen-Addad, Ruiquan Gao, Fabrizio Grandoni, Euiwoong Lee, and Ernest van Wijland. An improved greedy approximation for (metric) k-means. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025*, pages 233–240. IEEE, 2025. doi:10.1109/FOCS63196.2025.00016.

10 Vincent Cohen-Addad. A fast approximation scheme for low-dimensional k-means. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 430–440. SIAM, 2018. doi:10.1137/1.9781611975031.29.

11 Vincent Cohen-Addad, Arnaud de Mesmay, Eva Rotenberg, and Alan Roytman. The bane of low-dimensionality clustering. In *Proceedings of the 2018 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2018. doi:10.1137/1.9781611975031.30.

12 Vincent Cohen-Addad, Alessandro Epasto, Silvio Lattanzi, Vahab Mirrokni, Andres Muñoz Medina, David Saulpic, Chris Schwiegelshohn, and Sergei Vassilvitskii. Scalable differentially private clustering via hierarchically separated trees. In Aidong Zhang and Huzefa Rangwala, editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 221–230. ACM, 2022. doi:10.1145/3534678.3539409.

13 Vincent Cohen-Addad, Andreas Emil Feldmann, and David Saulpic. Near-linear time approximation schemes for clustering in doubling metrics. *J. ACM*, 68(6):44:1–44:34, 2021. doi:10.1145/3477541.

14 Vincent Cohen-Addad, Andreas Emil Feldmann, and David Saulpic. Near-linear time approximation schemes for clustering in doubling metrics. In *J. ACM*, volume 68, 2021. doi:10.1145/3477541.

15 Vincent Cohen-Addad, Fabrizio Grandoni, Euiwoong Lee, Chris Schwiegelshohn, and Ola Svensson. A $(2+\epsilon)$-approximation algorithm for metric k-median. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 615–624. ACM, 2025. doi:10.1145/3717823.3718299.

16 Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT approximations for k-median and k-means. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.42.

17 Vincent Cohen-Addad and Karthik C. S. Inapproximability of clustering in lp metrics. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 519–539. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00040.

18 Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. On approximability of clustering problems without candidate centers. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2635–2648. SIAM, 2021. doi:10.1137/1.9781611976465.156.

19 Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. Johnson coverage hypothesis: Inapproximability of k-means and k-median in ℓ_p-metrics. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1493–1530. SIAM, 2022. doi:10.1137/1.9781611977073.63.

20 Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in Euclidean and minor-free metrics. *SIAM J. Comput.*, 48(2):644–667, 2019. doi:10.1137/17M112717X.

21 Vincent Cohen-Addad, Silvio Lattanzi, Ashkan Norouzi-Fard, Christian Sohler, and Ola Svensson. Fast and accurate k-means++ via rejection sampling. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/babcff88f8be8c4795bd6f0f8cccca61-Abstract.html.

22 Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 169–182. ACM, 2021. doi:10.1145/3406325.3451022.

23 Sanjoy Dasgupta and Yoav Freund. Random projection trees for vector quantization. *IEEE Transactions on Information Theory*, 55(7):3229–3242, 2009. doi:10.1109/TIT.2009.2021326.

24 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for exponential-time-hypothesis-tight algorithms and lower bounds in geometric intersection graphs. *SIAM J. Comput.*, 49(6):1291–1331, 2020. doi:10.1137/20M1320870.

25 Irit Dinur. Mildly exponential reduction from gap 3sat to polynomial-gap label-cover. *Electron. Colloquium Comput. Complex.*, TR16-128, 2016. URL: https://eccc.weizmann.ac.il/report/2016/128.

26 Fedor V. Fomin, Petr A. Golovach, and Kirill Simonov. Parameterized k-clustering: Tractability island. *J. Comput. Syst. Sci.*, 117:50–74, 2021. doi:10.1016/J.JCSS.2020.10.005.

27 Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local search yields a PTAS for k-means in doubling metrics. *SIAM J. Comput.*, 48(2):452–480, 2019. doi:10.1137/17M1127181.

28 Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *J. Algorithms*, 31(1):228–248, 1999. doi:10.1006/jagm.1998.0993.

29 S Louis Hakimi. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations research*, 13(3):462–475, 1965.

30 Sariel Har-Peled. *Geometric approximation algorithms*. Number 173 in Mathematical Surveys and Monographs. American Mathematical Soc., 2011.

31 Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering (extended abstract). In *Proceedings of the Tenth Annual Symposium on Computational Geometry, Stony Brook, New York, USA, June 6-8, 1994*, pages 332–339, 1994. doi:10.1145/177424.178042.

32 Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 731–740, 2002. doi:10.1145/509907.510012.

33 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001. doi:10.1145/375827.375845.

34 Sándor Kisfaludi-Bak, Jesper Nederlof, and Karol Wegrzycki. A gap-ETH-tight approximation scheme for Euclidean TSP. *Journal of the ACM*, 72(6):1–48, 2025.

35 Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the Euclidean k-median problem. *SIAM J. Comput.*, 37(3):757–782, 2007. doi:10.1137/S0097539702404055.

36 Alfred A Kuehn and Michael J Hamburger. A heuristic program for locating warehouses. *Management science*, 9(4):643–666, 1963.

37 Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Inf. Process. Lett.*, 120:40–43, 2017. doi:10.1016/j.ipl.2016.11.009.

38 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. *SIAM J. Comput.*, 45(2):530–547, 2016. doi:10.1137/130938645.

39 Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. doi:10.1109/TIT.1982.1056489.

40 Meena Mahajan, Prajakta Nimbhorkar, and Kasturi R. Varadarajan. The planar k-means problem is np-hard. *Theor. Comput. Sci.*, 442:13–21, 2012. doi:10.1016/J.TCS.2010.05.034.

41 Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Performance of johnson-lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1027–1038, 2019. doi:10.1145/3313276.3316350.

42 Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense csps. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 78:1–78:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.78.

43 Jirí Matousek. On approximate geometric k-clustering. *Discret. Comput. Geom.*, 24(1):61–84, 2000. doi:10.1007/s004540010019.

44 Nimrod Megiddo and Kenneth J Supowit. On the complexity of some common geometric location problems. *SIAM journal on computing*, 13(1):182–196, 1984. doi:10.1137/0213014.

45 Satish B Rao and Warren D Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 540–550, 1998.

46 John Fred Stollsteimer. *The effect of technical change and output expansion on the optimum number, size, and location of pear marketing facilities in a California pear producing region*. University of California, Berkeley, 1961.

On Computing the (Exact) Fréchet Distance with a Frog

Jacobus Conradi
Department of Computer Science, University of Copenhagen, Denmark

Ivor van der Hoog
IT University of Copenhagen, Denmark

Eva Rotenberg
IT University of Copenhagen, Denmark

Abstract

The continuous Fréchet distance $\mathcal{D}_F(\pi, \sigma)$ between two polygonal curves π and σ is classically computed by exploring the *free space diagram* over the two curves. [SoCG'25] recently proposed a radically different approach: they approximate $\mathcal{D}_F(\pi, \sigma)$ by computing paths in a discrete graph that models a joint traversal of π and σ, recursively bisecting edges until the discrete distance converges to the continuous one. They implement their "frog-based" technique, and claim that it yields substantial practical speedups compared to the state-of-the-art implementations.

In this paper, we revisit this technique. We observe that, in its current form, it has three limitations: (i) it does not use exact arithmetic, (ii) its recursive bisection introduces the required monotonicity events to realise the Fréchet distance only in the limit, and (iii) it applies a heuristic simplification technique which is overly conservative. Motivated by theoretical interest, we develop new techniques that guarantee exactness, polynomial-time convergence and near-optimal lossless simplifications. We provide an open-source C++ implementation of our variant.

Our primary contribution is an extensive empirical evaluation on a broad, publically available, suite of real-world and synthetic data sets. Among the frog-based variants, exact computation indeed introduces overhead and increases median runtime. Yet, our new approach is often faster in the worst case, worst ten percent, or even the average runtime due to its worst-case convergence guarantees. More surprisingly, the implementation of [SoCG'19] dominates all frog-based implementations in performance – this finding contrasts previously published claims. These results provide a much-needed nuanced perspective on the capabilities and limitations of frog-based techniques: we showcase its theoretical appeal, but highlight its limited practical feasibility.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Algorithms engineering, Fréchet distance

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.35

Related Version *Full Version*: `https://arxiv.org/abs/2512.07728` [20]

Supplementary Material *Software (Source Code)*: `https://doi.org/10.5281/zenodo.19074030`

Funding *Ivor van der Hoog* and *Eva Rotenberg* are grateful to the Carlsberg Foundation for supporting this research via grant CF21-0302 "Graph Algorithms with Geometric Applications", and to the VILLUM Foundation for supporting this research via grant 37507 "Efficient Recomputations for Changeful Problems". *Jacobus Conradi* is supported by the Carlsberg Foundation, grant CF24-1929.

1 Introduction

The continuous Fréchet distance between two polygonal curves π and σ is classically described using the analogy of a person walking along π and a dog walking along σ, each moving continuously and monotonically. The continuous Fréchet distance $\mathcal{D}_F(\pi, \sigma)$ is the infimum

© Jacobus Conradi, Ivor van der Hoog, and Eva Rotenberg;
licensed under Creative Commons License CC-BY 4.0

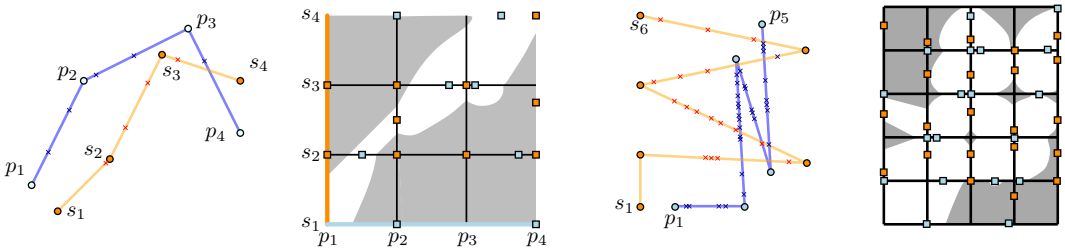


Figure 1 Two curves (π, σ) with their parameter space. White space indicates that the corresponding points are within some distance Δ. In each grid, we illustrate the edge events. For each monotonicity event defined by an edge e and vertices (x_1, x_2) we show the event on e with a cross.

over all such traversals of the maximum distance between them. A discrete analogue is defined similarly by imagining two frogs that hop monotonically along the vertices of π and σ, leading to the discrete Fréchet distance $\mathbb{D}_F(\pi, \sigma)$.

Let $\pi = (p_1, \ldots, p_n)$ and $\sigma = (s_1, \ldots, s_m)$. From an algorithmic and practical perspective, it is possible to decide whether their continuous or discrete decision problems is less than some input value Δ in an efficient manner. For a given value Δ, deciding whether $\mathbb{D}_F(\pi, \sigma) \leq \Delta$ reduces to determining whether there exists a monotone path in an $n \times m$ vertex-weighted grid where the weight of (p_i, s_j) is $d(p_i, s_j)$. Removing all vertices with weight greater than Δ and performing a graph search from (p_1, s_1) to (p_n, s_m) takes $O(nm)$ time. The continuous decision problem admits a related construction that we discuss in the preliminaries.

Computing the Fréchet distance and monotonicity events. Computing the *value* of the Fréchet distance is considerably more intricate. The continuous distance is realised by one of three types of events (see Figure 1). Vertex events correspond to distances between pairs of vertices in $\pi \times \sigma$; edge events correspond to distances between a vertex of one curve and an edge of the other; and monotonicity events correspond to triples (e, s, s'), where e is an edge of one curve, s, s' are vertices of the other curve, and the corresponding distance value is realised by the point on e minimising the maximum distance to s and s'. The discrete distance $\mathbb{D}_F(\pi, \sigma)$ is always realised by a vertex event. As there are $O(nm)$ such events one may therefore compute all distances, sort them, and apply the decision procedure, yielding an $O(nm \log nm)$ algorithm (or $O(nm)$ with a simple dynamic program). This version is also easy to make exact, as one can simply square the vertex-weights to maintain rational input. For the continuous setting, monotonicity events introduce two significant difficulties:

- There are $O(n^3 + m^3)$ such events, so no quadratic-time algorithm can enumerate them.
- Edge and monotonicity events involve equations that contain several square root monomials. They therefore offer no straightforward way to avoid numerical imprecision, even if the input is integral. This causes problems, not only for computing the distance value, but even for computing the correct combinatorial traversal that realises the Fréchet distance.

An exact decision procedure for the continuous variant was provided by [10], who use exact geometric predicates to test $\mathcal{D}_F(\pi, \sigma) \leq \Delta$ in $O(nm)$ time. Their implementation allows an approximation of $\mathcal{D}_F(\pi, \sigma)$ to b bits precision via binary search in $O(nm(\log nm + \log b))$ time, but it does not compute the exact value.

Approaching the continuous Fréchet distance with a frog. [26] recently proposed a different approach. Given (π, σ) they construct what they call the VE-graph whose vertices correspond to all vertex and edge events. In this graph, there is an edge between two vertices if they share

a cell and either the edge is monotone, or the events lie on an edge of the same curve (see Figure 2). The vertices are weighted by the event distance, and the shortest path from $(1, 1)$ to (n, m) in this graph corresponds to a (non-monotone) person-dog traversal γ. If γ happens to be xy-monotone, its continuous counterpart realises the Fréchet distance. Otherwise, non-monotonicities certify that the Fréchet distance is realised by a monotonicity event. Instead of computing monotonicity events, their algorithm bisects each edge of π and σ where γ is non-monotone in the corresponding row or column. This creates higher-complexity curves (π', σ') on which they recurse. They call this *refinement*, and its recursive application yields increasingly complex curves whose vertex and edge events approach the continuous Fréchet distance $\mathcal{D}_F(\pi, \sigma)$. In the limit, this procedure converges to introducing the necessary monotonicity events and thereby computes the continuous Fréchet distance through discrete shortest paths.

Their second contribution is a clever simplification technique that is agnostic to how the Fréchet distance between curves is computed. They show how to simplify two input curves (π, σ) into lower-complexity curves $(\overline{\pi}, \overline{\sigma})$ such that $\mathcal{D}_F(\pi, \sigma) = \mathcal{D}_F(\overline{\pi}, \overline{\sigma})$. The procedure is iterative: they first compute simplified curves $(\overline{\pi}, \overline{\sigma})$ and evaluate $\mathcal{D}_F(\overline{\pi}, \overline{\sigma})$ using any method. This yields a person-dog traversal, from which they generate a witness certifying whether $\mathcal{D}_F(\overline{\pi}, \overline{\sigma}) = \mathcal{D}_F(\pi, \sigma)$ holds. If not, the curves are partially un-simplified, reintroducing complexity until the criterion is satisfied. Based on our observations, this simplification step is likely the main source of the speedy performance of their implementation.

Applications of Fréchet distance. The Fréchet distance has many well-known applications, in particular in the analysis and visualisation of movement data [11, 15, 29, 37], and has enjoyed continued interest in many different settings [1, 2, 7, 17–19, 23]. It is a versatile measure used across a wide range of domains, including handwriting [33], coastlines [30], geographic outlines [21], trajectories of vehicles, animals, and sports players [9, 15, 32, 34], air traffic [8], and protein structures [28]. These typically provide their own Fréchet distance implementations. A recent and particularly active domain is *trajectory clustering*, where the task is to group (sub)curves into clusters such that each cluster has low pairwise Fréchet distance. Several approaches adopt this principle, including the subtrajectory clustering algorithm of [3], the map construction and clustering algorithms [11, 12], the clustering method of [13] (implemented in the `MOVETK` library [24]), the clustering algorithm of [16], and the clustering of [35]. In the *vast* majority of these examples, the implementation computes the discrete Fréchet distance, as this is considerably easier to evaluate in practice.

Two of these approaches [11, 12] compute the *semi-weak* Fréchet distance, in which the person and the dog are not required to be monotone within an edge. This variant corresponds to the edge-monotone minimum-cost traversal computed from [26] before their curve refinement procedure is applied. These examples illustrate the need for efficient and adaptable Fréchet distance implementations, and highlight why the refinement-based approach of [26] is appealing: it suggests the possibility of computing the exact continuous Fréchet distance through a discrete traversal mechanism that is compatible with existing algorithmic paradigms and interfaces.

Contribution. We re-examine the frog-based framework of [26], as we improve three aspects of their contribution.

Contribution 1: exact-value computations. We find the overarching method conceptually very interesting (and even elegant), however, their algorithm does not compute the exact Fréchet distance. They state that "as the calculation of the optimal Fréchet distance involves

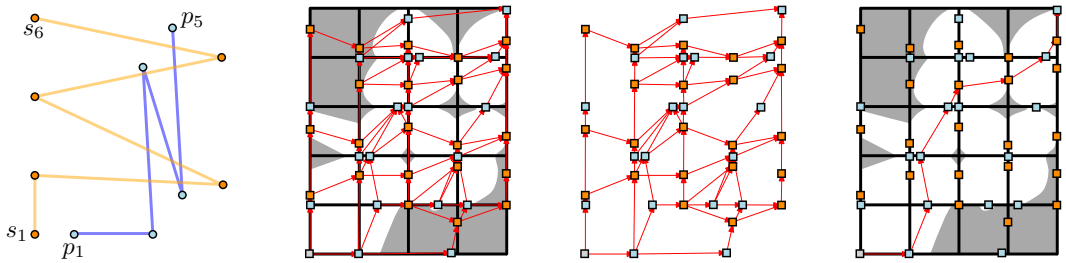


Figure 2 Two pairs of curves (π, σ) and their VE-graph which is a directed and vertex-weighted graph. The minimum-cost path from $(1, 1)$ to (n, m) in this graph is not a monotone traversal.

distances, impreciseness is unavoidable". This is not necessarily true as there exist exact-evaluation predicates. In theory, one can even assume a real RAM. Their current approach would remain imprecise under such assumptions:

1. Their implementation inherently uses floating-point arithmetic and thus imprecision.
2. However, even with exact real-valued arithmetic, their edge bisection does not introduce the monotonicity events that realise the exact Fréchet distance. This refinement converges to the correct value only in the limit, with no bound on the number of required iterations.

Their lossless simplification step is theoretically sound and would yield the exact Fréchet distance under real-valued arithmetic. However, the procedure is somewhat crude: to ensure that the Fréchet distance remains unchanged after simplification, they impose overly conservative constraints on how the curves may be simplified. The original paper briefly acknowledges the possibility of an exact algorithm, but quickly states that "from implementation point of view such a modification seems pointless, and we did not pursue it any further". Our motivation is to understand to what extent this framework *can* be made exact and efficient, and to provide the required components to do so.

1. We explain in detail how to unpack square-root expressions for exact evaluation. This is non-trivial, particularly for monotonicity events and curve simplifications where several square-root terms must be handled simultaneously.
2. We replace their heuristic refinement procedure with a new output-sensitive algorithm that generates the required monotonicity events on demand. As a consequence, we obtain provable polynomial-time convergence, even under real-valued computations.
3. We strengthen the lossless simplification technique. Whilst the method from [26] works as a black-box given any Fréchet computation algorithm that outputs a traversal, we note that this frog-based paradigm allows for a tighter algorithm and analysis. This leads to an alternative and new algorithm for lossless simplification under the frog-based method.

Contribution 2: exact implementation. We provide an open-source C++ implementation of our adapted algorithm with our exact arithmetic and monotonicity event-generation. This thereby is the first exact implementation. We experientially compare our solution to the one provided in [26]. Unsurprisingly, our median running times increase due to exact computation. Our exact-value representations simply do not allow for vectorisation, which comes at performance cost. Somewhat unexpectedly, the worst-case running time drastically improves. We can show instances where the recursive refinement encounters its worst-case behaviour and terminates with an approximation only after many refinement steps. For some data sets, the worst-case behaviour is such that even our average time is an improvement.

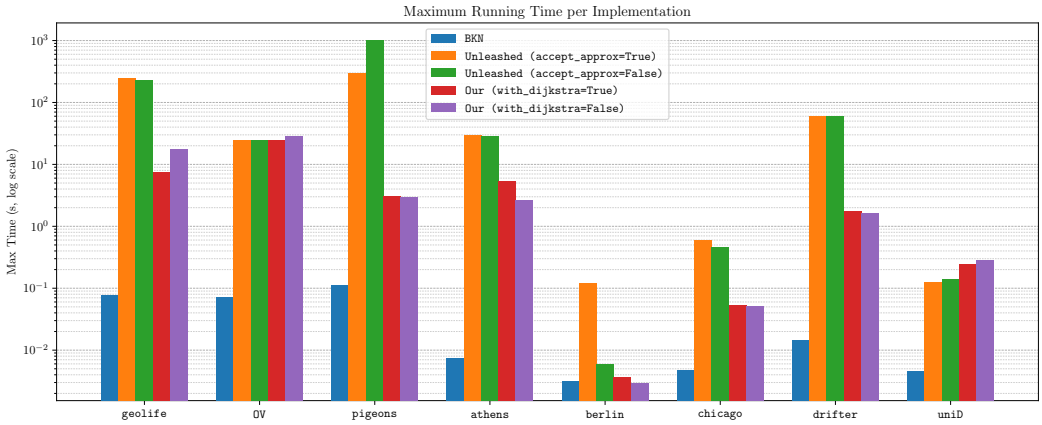


Figure 3 Maximum running times in seconds, log-scale, for curves from eight data sets.

Contribution 3: extensive empirical analysis. Our final contribution is the most significant. The original publication [26] claims that their running times are significantly faster than previous Fréchet computations. Across all our data sets, we observe a substantially different performance profile. In particular, the highly-optimised implementation of Bringmann, Künnemann, and Nusser `BKN` [10] dominates all frog-based implementations in performance (e.g., Figure 3). As a consequence of our discovery, the authors of [26] posted on their ArXiV version an errata explaining that their experimental setup was flawed.

At first sight, our empirical analysis may appear unfavourable to our contribution, as it shows that on many practical instances the performance of this new frog-based approach is surpassed by classical techniques. However, we argue for our significance: we provide the first exact Fréchet distance implementation. Moreover, the empirical analysis reported at SoCG'25 [26] claimed strong practical performance over [10]. Our broader evaluation provides a necessary complementary picture, contradicting this claim. We view this contrast as valuable to our community: it clarifies that this technique is valuable, but also clearly shows the limits of its use. Ultimately, we argue that the frog-based technique of [26] is of genuine theoretical interest, as it can be augmented to compute the exact Fréchet distance in a unique manner with provable guarantees. At the same time, its practical use is debatable.

2 Preliminaries

The input consists of polygonal curves π and σ. A *curve* $\pi = (p_1, \ldots, p_n)$ is an ordered sequence of *vertices* connected by straight-line *edges*. We implicitly endow every n-vertex curve π with a bijective, monotone parametrisation $\pi : [1, n] \to \mathbb{R}^d$ such that $\pi(i) = p_i$ for all integers i. For $t \in [i, i+1]$, the mapping is defined by linear interpolation $\pi(t) := p_i + (p_{i+1} - p_i)\,(t - i)$.

For an ordered pair of curves (π, σ) of sizes (n, m), a *traversal* is any pair of continuous, monotone mappings $\alpha : [0, 1] \to [1, n]$ and $\beta : [0, 1] \to [1, m]$. The (continuous) Fréchet distance $\mathcal{D}_F(\pi, \sigma)$ is defined as $\mathcal{D}_F(\pi, \sigma) := \inf_{\text{traversals } (\alpha,\beta)} \max_{t\in[0,1]} d\big(\pi(\alpha(t)), \sigma(\beta(t))\big)$.

Equivalently, consider the parameter space $\Gamma_{\pi,\sigma} := [1, n] \times [1, m]$, subdivided into an $n \times m$ grid. A traversal corresponds to any continuous, xy-monotone curve $\gamma(t) = (\alpha(t), \beta(t))$ from $(1, 1)$ to (n, m) in $\Gamma_{\pi,\sigma}$. We therefore freely consider traversals as monotone curves.

Deciding. We can decide whether $\mathcal{D}_F(\pi,\sigma) \leq \Delta$ as follows. The area $\{(a,b) \in [1,n]\times[1,m] \mid d(\pi(a),\sigma(b)) \leq \Delta\}$ is the *free space*. Via DFS over the free space we decide in $O(nm)$ time whether the free space contains a monotone curve from $(1,1)$ to (n,m), i.e., $\mathcal{D}_F(\pi,\sigma) \leq \Delta$ [5].

Computing. The Fréchet distance is realised by one of three types of *events*:

- *Vertex events*: distances between pairs of vertices in $\pi \times \sigma$.
- *Edge events*: distances between a vertex of one curve and an edge of the other.
- *Monotonicity events*: triples (e,s,s') where e is an edge of one curve and s,s' are vertices of the other, corresponding to minimising the maximum distance to s and s'.

A naive algorithm enumerates all events in $O(nm^2+mn^2)$ time, sorts them, and performs a binary search with the decision algorithm to obtain $\mathcal{D}_F(\pi,\sigma)$. [5] observed that many monotonicity events are irrelevant and achieved an $O(nm\log nm)$-time exact algorithm. [25] gave a considerably simpler approach: as Δ increases, connected components of the free space merge, and a *critical value* is the smallest Δ at which a new merge occurs. Each cell has one connected component, implying $O(nm)$ critical values. By comparing adjacent cells, all critical values can be generated in $O(nm\log nm)$ time. The fastest known algorithm is by [14], running in $O\big(n^2\sqrt{\log n}\,(\operatorname{loglog} n)^{3/2}\big)$ time.

Retractability. For curves (π,σ) and a minimum-cost traversal $\gamma = \{\gamma(t) \mid t \in [0,1]\}$, define $\operatorname{cost}(\gamma) := \max_{(a,b)\in\gamma} d\big(\pi(a),\sigma(b)\big)$. Let $(a,b) \in \gamma$ be a point that attains this maximum. Then γ decomposes into two open curves: γ_1 from $(1,1)$ to (a,b) and γ_2 from (a,b) to (n,m). [26] call γ *retractable* if γ_1 and γ_2 are minimum-cost monotone curves for their respective subcurves, and if γ_1 and γ_2 are themselves retractable (Figure 4). Computing the Fréchet distance using an adaption of Dijkstra yields a retractable traversal.

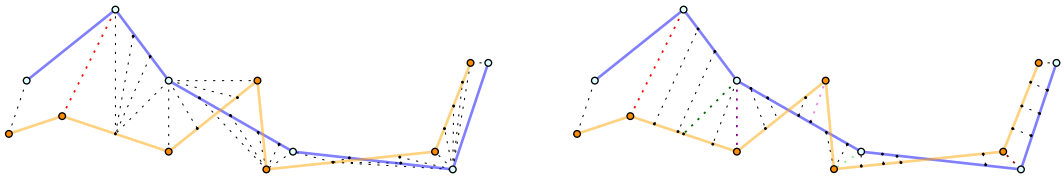


Figure 4 (Left) A traversal between two curves where the red leash realises $\mathcal{D}_F(\pi,\sigma)$. The traversal may be "lazy": a large Fréchet distance allows long pauses while the other curve progresses. (Right) A retractable traversal achieves progressively smaller bottleneck leash lengths.

3 The frog-based approach

We paraphrase and explain the frog-based approach from [26], defining their VE-graph slightly differently to reduce the number of edge-cases the definition incurs. We then detail how to make it converge in a polynomial number of rounds, even on a real-RAM, by creating an algorithm that generates the required monotonicity events on the fly.

▶ **Definition 1.** *Each edge e of the grid $\Gamma_{\pi,\sigma}$ has an edge event defining a point that we call an* edge-distance yardstick *(or* eddy*). $V(\pi,\sigma)$ denotes the set of all grid vertices and eddys.*

▶ **Definition 2.** *The VE-graph is a directed vertex-weighted graph on $V(\pi,\sigma)$ where the weight of each vertex is the distance between the pairs of points corresponding to the vertex. For each cell of $\Gamma_{\pi,\sigma}$, consider its four corners and eddys. We connect these points in an xy-monotone manner. In addition, the bottom horizontal eddy has a directed edge to the top horizontal eddy, and the left vertical eddy has a directed edge to the right vertical eddy; see Fig. 5.*

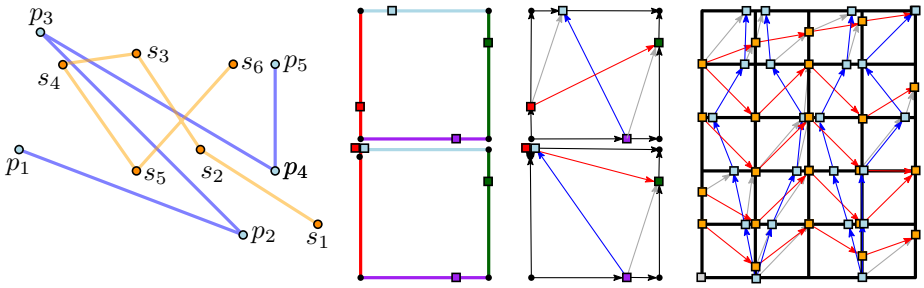


Figure 5 The VE graph considers for each cell the four corners and the four eddys. We include all monotone edges between these vertices. Edges connecting the eddys between horizontal (or vertical) edges may be non-monotone. If vertices coincide, then the picture gets a bit messy.

▶ **Definition 3.** *A path in the VE-graph is VE-respecting if it runs from* $(1,1)$ *to* (n,m) *and, within every row and every column, its restriction to the row or column is connected.*

$$\mathcal{V}_E(\pi,\sigma) := \min_{\text{VE-respecting path } P} \max_{\text{vertex } v \in P} \text{weight}(v). \qquad \text{Observe that } \mathcal{V}_E(\pi,\sigma) \leq \mathcal{D}_F(\pi,\sigma).$$

From a path realizing the VE distance, one may also derive an upper bound on $\mathcal{D}_F(\pi,\sigma)$.

▶ **Definition 4.** *Let* P *be a path realizing* $\mathcal{V}_E(\pi,\sigma)$ *then* P *is a curve. One can greedily morph* P *into a traversal by ensuring that one never decreases in either coordinate. This defines an* xy*-monotone traversal* $\gamma(t) = (\alpha(t), \beta(t))$*. The* Interpolated VE distance *is then defined as:*

$$\mathcal{I}_{VE}(\pi,\sigma) := \max_{t\in[0,1]} d\big(\pi(\alpha(t)), \sigma(\beta(t))\big). \qquad \text{Observe that } \mathcal{D}_F(\pi,\sigma) \leq \mathcal{I}_{VE}(\pi,\sigma).$$

The *elevation function* maps $(a,b) \in \Gamma_{\pi,\sigma}$ to $d(\pi(a),\sigma(b))$. The sublevel sets of this function are clipped ellipses, whose minima on each cell edge occur precisely at the corresponding eddys. It follows that $\mathcal{V}_E(\pi,\sigma) = \mathcal{D}_F(\pi,\sigma) = \mathcal{I}_{VE}(\pi,\sigma)$ whenever the path P realizing $\mathcal{V}_E(\pi,\sigma)$ is monotone. This is the core of the frog-based refinement approach (Algorithm 1).

Algorithm 1 Refinement Procedure from [26].

1: Compute the VE-graph, the value $\mathcal{V}_E(\pi,\sigma)$, and the path P realising it
2: **if** P is monotone **then** $\mathcal{V}_E(\pi,\sigma) = \mathcal{D}_F(\pi,\sigma)$ so **return** $\mathcal{V}_E(\pi,\sigma)$
3: Otherwise, insert vertices on π for every non-x-monotone column, and vertices on σ for every non-y-monotone row. Then recurse on the refined curves (π,σ).

3.1 The approach in [26]: bisecting and convergence

Har-Peled, Raichel, and Robson [26] execute step 3 of Algorithm 1 in a heuristic manner. Let P be a path realising $\mathcal{V}_E(\pi,\sigma)$ for the current input (π,σ). If P is non-monotone in a column that corresponds to an edge e of π then they introduce a vertex half-way e (likewise, they bisect edges of σ for each row where P is non-monotone). Thus, given P they bisect some edges and create higher-complexity curves (π,σ) on which they recurse.

Recall that each monotonicity event (e,p,p') corresponds to a point on the edge e; we refer to such a point as a *monotonicity vertex*. Let (π',σ') be the curves obtained from (π,σ) by inserting, on π and σ respectively, a vertex at every monotonicity vertex. It is observed in [26] that the path P realising $\mathcal{V}_E(\pi',\sigma')$ is xy-monotone, and therefore $\mathcal{V}_E(\pi',\sigma') = \mathcal{D}_F(\pi,\sigma)$. Under exact-value arithmetic, the recursive halving procedure of [26] inserts in the limit all monotonicity vertices of π and σ, and thus converges to the exact Fréchet distance $\mathcal{D}_F(\pi,\sigma)$.

3.2 Our approach and rate of convergence

In [26] they use floating-point arithmetic. Therefore, the computation is non-exact. Moreover, bisecting edges will typically not generate the exact monotonicity vertices. They remark that under exact computations one can compute and introduce monotonicity vertices instead, but do not provide details, and naively there are cubicly many such events. Floating-point arithmetic is not unavoidable. Many libraries support exact arithmetic (e.g., CGAL). We engineer a solution that alters step 3 to provide guarantees for exactness and convergence: Let the path P realising $\mathcal{V}_E(\pi, \sigma)$ be not x-monotone in a column C corresponding to an edge e. Let the subpath of P restricted to C be from a to b. We execute step 3 by computing the minimum number of monotonicity vertices that need to be introduced on e such that, next iteration, the min-cost path from a to b is monotone. Our algorithm is inspired by the critical values from [5, 25] from the preliminaries, but ultimately has a different objective.

On efficiency. Exact-value arithmetic inevitably incurs overhead. Evaluating exact geometric predicates is substantially slower than using floating-point arithmetic. Moreover, the resulting values must be maintained as abstract algebraic objects rather than bit-level representations, preventing efficient vectorisation. This is partly why the algorithm from [10] does not use exact arithmetic even though it is integrated in CGAL. We therefore expect exact computation to increase median running times when we introduce exact arithmetic. To mitigate this effect, we restrict the input to 32-bit integers, as this allows the most frequent operation – the computation of dot products – to still be efficient. For our exact arithmetic, based on the computed dot products, we discuss in the full version our own integer-based kernel, to remain practically somewhat competitive [20]. Importantly, our algorithm does not rely on integrality assumptions: it applies unchanged under any exact-evaluation kernel.

High-level overview. In the full version [20] we develop the following algorithm: the input is a column C where P is not x-monotone, corresponding to an edge e of π. Let P, restricted to C, go from a to b. We create two new cells by extending a horizontal segment from a and b. We discard all cells of C that do not intersect P. We index the remaining cells as $C_1, \cdot, C_k$ from bottom to top, which induce horizontal segments $h_1, \ldots, h_{k+1}$. Imagine Δ increasing from 0 to ∞. For each value of Δ and each pair (i, j) with $i < j$, h_i can Δ-*reach* h_j if there exists a monotone path from a point on h_i to a point on h_j in the Δ free space (see Figure 6).

A value Δ is an *join event* if there exist (i, j) such that h_i can Δ-reach h_j from Δ onward. In this case, h_i can also Δ-reach all $h_{j'}$ with $j' \in [i, j)$. We compute the minimum number of join events to realise a monotone path from h_1 to h_k using a linked–list of buckets.

The horizontal segments are partitioned into buckets, each forming a vertical interval $[i, j]$ such that h_i can Δ-reach h_j. We initialise Δ as 0, and every horizontal segment forms its own bucket. We then merge buckets in a bottom-up fashion, maintaining the minimum Δ required such that h_1 can Δ-reach h_i for increasing i. Between $(h_1, \ldots, h_i)$ and h_{i+1}, we compute the smallest value $\Delta' \geq \Delta'$ for h_1 can Δ-reach h_{i+1} using a min-heap that contains join events and other events that indicate for which pair $(j, i + 1)$ the join event is relevant.

Whenever we pop the next event, we update Δ. If the updated Δ allows us to Δ-reach h_{i+1} from h_1 then we merge h_{i+1} into the bucket. We then keep incrementing i until h_1 cannot Δ-reach h_{i+1}, at which point we continue the event-based approach. Whenever we merge h_k into the bottom bucket, h_1 can Δ-reach h_k. When all buckets are joined, we have all join events required for a monotone path from h_1 to reach h_k. From this, we obtain the minimum number of monotonicity vertices needed so that, in the next iteration, the min-cost path from a to b is monotone. Doing this for all non-monotone rows and columns implies:

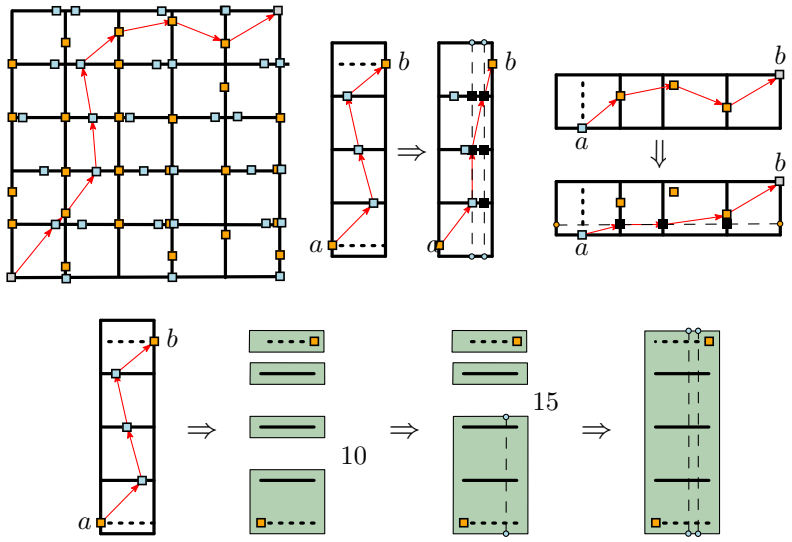


Figure 6 The input is a non-monotone path P in the VE-graph. We identify all rows and columns where P is non-monotone. For a column, we isolate the horizontal segments and bucket them. Between buckets, we store the minimum leash at which a monotone path from bottom to top exists.

▶ **Theorem 5.** *Let (π, σ) be the original input curves with n and m vertices. Algorithm 1 can be supported by a data structure using $O(n + m + M)$ space, where $M \in O(n^3 + m^3)$ is the number of introduced monotonicity vertices. Each recursive call runs in $O(nm + M)$ time for Step 1 and $O((n + m + M) \log^2(n + m + M))$ time for Step 3. Under exact-value arithmetic, the procedure returns the exact Fréchet distance after $O(n^3 + m^3)$ recursive calls.*

4 Lossless Fréchet distance simplification

The second (perhaps even main) contribution of [26] is a black-box lossless simplification algorithm. Computing $\mathcal{D}_F(\pi, \sigma)$ for large complexity curves π and σ is an expensive operation. It would be preferable if we would have access to some lower-complexity curves $(\overline{\pi}, \overline{\sigma})$ where $\mathcal{D}_F(\pi, \sigma) = \mathcal{D}_F(\overline{\pi}, \overline{\sigma})$. Then we could simply operate on these lower-complexity curves to output the distance. Har-Peled, Raichel, and Robson [26] can, for any Fréchet computation algorithm, achieve this using any *vertex-restricted* curve simplification:

▶ **Definition 6.** *We define for π a (vertex-restricted) simplification $\overline{\pi}$ as any subset of vertices in π, ordered by their index. For any $\mu \in \mathbb{R}_{\geq 0}$ we say $\overline{\pi}$ is a μ-simplification if $\mathcal{D}_F(\pi, \overline{\pi}) \leq \mu$.*

The curve simplification used in [26] is some heuristic combination of techniques from [6, 22].

Naïvely, one can $(1 + \varepsilon)$-approximate $\mathcal{D}_F(\pi, \sigma)$ with simplifications by fixing some ε and μ. Compute μ-simplifications $(\overline{\pi}, \overline{\sigma})$ and set $M = \mathcal{D}_F(\overline{\pi}, \overline{\sigma})$. By the triangle inequality, $M - 2\mu \leq \mathcal{D}_F(\pi, \sigma) \leq M + 2\mu$. If $\mu \leq \varepsilon M$, then M $(1 + \varepsilon)$-approximates $\mathcal{D}_F(\pi, \sigma)$; otherwise μ is decreased cleverly until we terminate after $O(\log_{1+\varepsilon} n)$ rounds [22].

[26] avoids this approximation step by exploiting triangle inequalities more directly. Let $(\overline{\pi}, \overline{\sigma})$ be any vertex-restricted simplification of (π, σ) and consider the traversals $\gamma_0(t)$, $\gamma_1(t)$, and $\gamma_2(t)$ realising $\mathcal{D}_F(\pi, \overline{\pi})$, $\mathcal{D}_F(\overline{\pi}, \overline{\sigma})$, and $\mathcal{D}_F(\overline{\sigma}, \sigma)$, respectively. They combine these traversals in linear time into a traversal $\gamma(t) = (\alpha(t), \beta(t))$ between (π, σ) whose maximum

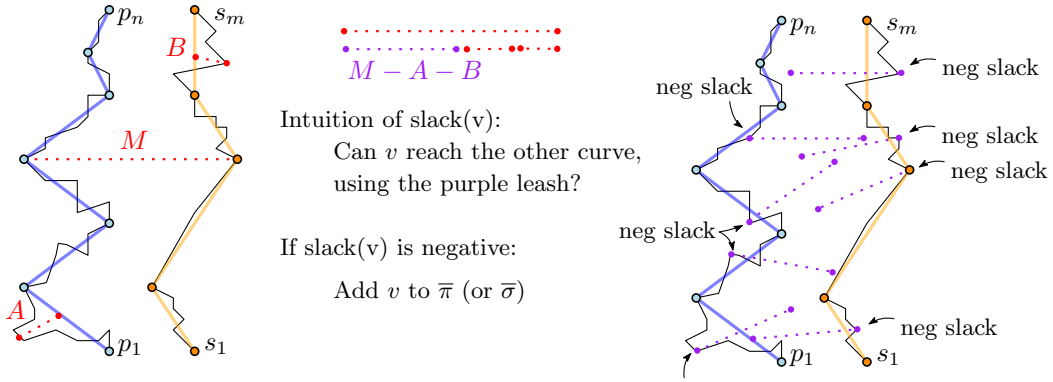


Figure 7 Curves (π, σ) and their simplifications. The value $LB(\overline{\pi}, \overline{\sigma})$ is the maximum leash length M between $(\overline{\pi}, \overline{\sigma})$, minus the maximum leash lengths A and B between $(\pi, \overline{\pi})$ and $(\sigma, \overline{\sigma})$.

leash length gives an upper bound on $\mathcal{D}_F(\pi, \sigma)$. Conversely, $\mathcal{D}_F(\pi, \sigma)$ is at least $\mathcal{D}_F(\overline{\pi}, \overline{\sigma})$ minus the Fréchet distances from each curve to its simplification, and we get the following:

$$\mathcal{D}_F(\overline{\pi}, \overline{\sigma}) - \mathcal{D}_F(\pi, \overline{\pi}) - \mathcal{D}_F(\sigma, \overline{\sigma}) \;\leq\; \mathcal{D}_F(\pi, \sigma) \;\leq\; \max_{t \in [0,1]} d(\pi(\alpha(t)), \sigma(\beta(t))).$$

Computing $\mathcal{D}_F(\pi, \overline{\pi})$ and $\mathcal{D}_F(\sigma, \overline{\sigma})$ is costly, so they compute greedy traversals $\gamma_0^*(t)$ and $\gamma_2^*(t)$ that overestimate these values instead, yielding $\mathcal{D}_F^*(\pi, \overline{\pi})$ and $\mathcal{D}_F^*(\sigma, \overline{\sigma})$. Combined, these give a greedy traversal $\gamma^*(t) = (\alpha^*(t), \beta^*(t))$ that over-estimates $\mathcal{D}_F(\pi, \sigma)$ and leads to

$$LB(\bar{\pi}, \bar{\sigma}) := \mathcal{D}_F(\bar{\pi}, \bar{\sigma}) - \mathcal{D}_F^*(\pi, \bar{\pi}) - \mathcal{D}_F^*(\sigma, \bar{\sigma}) \leq \mathcal{D}_F(\pi, \sigma) \leq \max_{t \in [0,1]} d(\pi(\alpha^*(t)), \sigma(\beta^*(t))). \quad (1)$$

Introducing slack. Given $LB(\overline{\pi}, \overline{\sigma})$ and $\gamma^*(t)$, each curve vertex v receives a *slack* $\texttt{slack}(v) := LB(\overline{\pi}, \overline{\sigma}) - g_v$, where g_v is the maximum leash length attained at v along $\gamma^*(t)$ (Figure 7). Negative slack marks v as a bottleneck. Each vertex p of $\overline{\pi}$ or $\overline{\sigma}$ inherits the minimum slack of all original vertices between p and the next vertex. Slack is then heuristically propagated: every vertex adopts the minimum slack among its eight closest neighbours along $\overline{\pi}$ or $\overline{\sigma}$. The slack of an edge is the minimum slack of its endpoints. They then perform the following:

> while there exists a negative-slack edge e on $\overline{\pi}$ (resp. $\overline{\sigma}$), and the edge e is not an edge of π (resp. σ), un-simplify the edge e (reintroducing missing vertices).

By construction of slack, its propagation, and (1), the process ends with $\mathcal{D}_F(\pi, \sigma) = \mathcal{D}_F(\overline{\pi}, \overline{\sigma})$.

Our contribution. In the full version, we observe that the above technique, while elegant, is overly conservative and somewhat coarse [20]. We refine it in two ways.

(i) Tighter bounds. Their lower bound uses the global maximum leash length M between $(\overline{\pi}, \overline{\sigma})$, minus the global maxima A and B between $(\pi, \overline{\pi})$ and $(\sigma, \overline{\sigma})$, respectively. When M, A and B are realised at unrelated parts of the traversal (see Figure 7), this yields an unnecessarily small value $LB(\overline{\pi}, \overline{\sigma}) = M - A - B$. This in turn causes more vertices than needed to trigger unsimplification. Using the frog-based framework, we obtain a sharper bound (Figure 8). For each edge of $\overline{\pi}$ we compute the maximum leash length between that edge and π, and assign this as a negative weight. In the VE-graph, the eddys on this edge add this negative weight to their own weight. The min-cost path in this reweighted graph

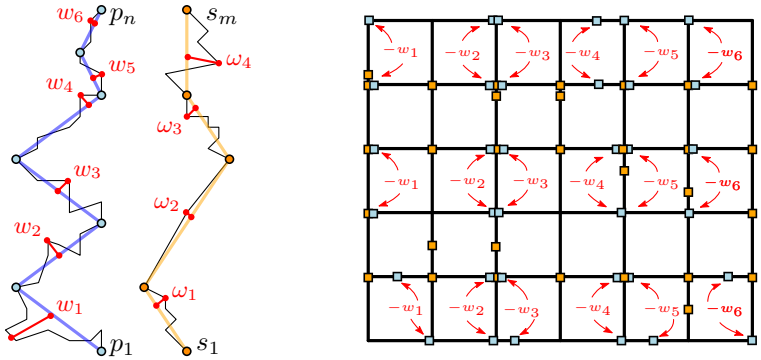


Figure 8 Given $(\pi, \overline{\pi})$ we can give a weight to each edge of $\overline{\pi}$. Each edge over $\overline{\pi}$ is a column in the parameter space, in which we apply its negative weight on the corresponding eddys.

yields a lower bound for $\mathcal{D}_F(\pi, \sigma)$ by the same triangle-inequality argument as in [26], but now based on local rather than global maxima. For many instances, this increases the lower bound and therefore the slack of vertices – causing our algorithm to unsimplify fewer edges.

(ii) A tighter slack propagation rule. Their propagation step takes the minimum slack over each block of eight consecutive vertices. This removes any edge cases for applying the slack, ensuring their correctness. However, this method is also rather crude and causes unnecessary unsimplification. We tighten this step by propagating slack along at most two edges per vertex, avoiding this coarse eight-neighbour spread.

5 Experiments and empirical analysis

Our primary contribution is a C++ implementation of our exact variant of the frog-based technique, together with an extensive empirical evaluation. We provide two implementations: `Exact_Dijkstra`, which computes the path P in the `VE`-graph using Dijkstra's algorithm in a manner similar to [26] and yields a traversal with retractability properties; and `Exact_Sweepline`, which instead employs a sweepline to compute P, guaranteeing linear space usage. We compare our implementations to `Unleashed`: the Julia code of [26], which is their fastest version. To obtain the most accurate output from `Unleashed`, we invoke `frechet_c_compute` with `accept_approximation=false` (this still approximates). We also include a version with approximation parameters enabled. Finally, we include the `BKN` algorithm of [10], the fastest known alternative implementation (which is also inexact). We do not use its CGAL variant, as it is slower, while still being inexact.

Experiments. The analysis in [26] uses three animal-tracking data sets: `Geolife`, `Birds`, and `Pigeons`. These trajectories are goal-oriented. Their implementation benefits from this in one major way compared to [10]: as a simplification is applied, this greatly reduces complexity. Consequently, they operate on significantly smaller effective input.

The lossless simplification of [26] applies to any Fréchet algorithm, including [10]. We *could* enable a direct comparison by applying it to [10]. Yet, we deliberately refrain from doing so: the simplification incurs overhead, and [10] is so fast that this may actually slow it down. Despite not using simplifications, our experiments will show that [10] dominates the frog-based approaches, removing the need to improve it further. All experiments are single-threaded and executed on an AMD Ryzen 5 7600X (4.7 GHz) with 96 GB RAM.

Table 1 Overview of all data sets used in our experiments.

Name	Real world	# curves	total # vertices	median # vertices	# of curves compared
`Geolife`	yes	18671	24895648	506	300 choose 2
`Pigeons`	yes	539	6229319	7502	300 choose 2
`Athens`	yes	120	72439	662	120 choose 2
OV	no	220	39060	130	110 (already pairs)
`Chicago`	yes	888	118360	121	100 choose 2
`Berlin`	yes	27188	192223	7	100 choose 2
`Drifter`	yes	2012	1792084	997	100 choose 2
`UniD`	yes	362	214076	495	100 choose 2

Data. Our real-world data sets are summarised in Table 1. The first two (`Geolife`, `Pigeons`) are from [26]. We additionally include `Athens`, `Chicago`, and `Berlin` GPS traffic data sets, widely used in Fréchet benchmarks [4, 11, 27, 36]. We include the `Drifter` data set of [16], containing drifting-buoy trajectories. We add the UnID data set from [31] which is a traffic dataset captured on German highways. Except for `Drifter`, these sets are goal-oriented with minimal backtracking; lossless simplification therefore removes relatively many vertices, which favours the frog-based approaches. Finally, we include a set `OV`[1] of difficult synthetic pairs of curves that are derived from Orthogonal Vectors instances [10].

The `Geolife` data set alone induces over 170,000,000 pairs of curves, so we cannot possibly compute the Fréchet distance between all pairs of curves. We follow [26] and subsample. In our case, we compare all 45,000 pairs among a subselection of 300 curves from the `Geolife` and `Pigeons` data sets. We further augment the analysis by comparing all curves from the `Athens` data set and from the adversarial `OV` data set. To broaden our evaluation, we also include the remaining real-world data sets, where, due to time constraints, we subsample curves and compare all pairs among them. The experiments are primarily bottlenecked by the `Unleashed` implementation[2], and the full evaluation requires several days to complete. Our results indicate no substantial qualitative differences between sampling 100×100 or 300×300 curves from the same data.

Criteria. Our primary evaluation metric is running time (including file-reading time). For exactness, our approach is the most accurate if the input coordinates fit within a 32-bit integer, followed closely by `BKN`; Table 2 briefly reports accuracies for the data sets satisfying this condition. We could compare across all data by truncating all input to 32-bit coordinates, but we refrain from doing so since `BKN` can handle this data natively (albeit inexact).

5.1 Comparing frog-based approaches

We first compare our frog-based implementations to `Unleashed`. This comparison is motivated mainly by scientific curiosity: exact-value computation is more difficult than computing an approximation, so one naturally expects a performance penalty. Our interest is in quantifying this penalty – in other words, in understanding the practical cost

[1] Obtained via personal communication with one author from [10], available with our source code
[2] A time limit of 5 minutes was hit multiple times

Table 2 Accuracy errors, obtained by converting exact square root distances to a `double`.

Data set	Max	Mean	Median
		`Exact`	
`OV`	0	0	0
`athens`	0	0	0
`UnID`	0	0	0
		`BKN`	
`OV`	$3.039939 \cdot 10^{-13}$	$2.756980 \cdot 10^{-13}$	$3.039939 \cdot 10^{-13}$
`athens`	$9.337306 \cdot 10^{-10}$	$5.865283 \cdot 10^{-12}$	$1.269745 \cdot 10^{-12}$
`UnID`	$9.337306 \cdot 10^{-10}$	$5.865283 \cdot 10^{-12}$	$1.269745 \cdot 10^{-12}$
		`Unleashed`	
`OV`	$4.499666 \cdot 10^{-6}$	$1.430129 \cdot 10^{-6}$	$1.181895 \cdot 10^{-16}$
`athens`	$9.267924 \cdot 10^{-6}$	$1.948025 \cdot 10^{-7}$	$9.974841 \cdot 10^{-15}$
`UnID`	$9.337306 \cdot 10^{-10}$	$5.865283 \cdot 10^{-12}$	$1.269745 \cdot 10^{-12}$

of exactness. Table 3 compares the efficacy of the `Unleashed` implementation versus `Exact_Sweepline` (the comparison remains roughly the same when we set for `Unleashed` the boolean `accept_approximation=true`, or when we compare to `Exact_Dijkstra` instead).

As expected, the median running time of our exact variants deteriorates. However, we note that our exact approach has considerably better worst-case performance, even on this advantageous real-world data. On some instances, `Unleashed` is a factor 100 slower. Whenever our worst-case is slower, it is not slower by much. For data sets where curves have sizable complexity, our method becomes the preferred method from the 90th percentile onwards. On the mapconstruction data sets, the median number of vertices is very small. The OV data set is a difficult instance where our exact computations are extremely costly. We note that except for worst-case outliers, the approaches are relatively competitive. We think that this data nicely represent the cost of exact value computation, and the benefit of worst-case guarantees for runtime.

5.2 Comparing Fréchet distance computations

We now provide a comparison between our exact frog-based method, the approach of [26], and the state-of-the-art `BKN` implementation [10]. In this wider evaluation, we consider two configurations of `Unleashed`: the version with `accept_approximation=false` (which, still returns an approximation) and the supposedly faster configuration that sets `accept_approximation=true`. Our exact implementations use either Dijkstra or a sweepline.

5.3 Results

Table 4 shows the full set of experimental results. These results highlight the volatility of the frog-based approach: the standard deviation of these approaches is considerably larger than `BKN` (although lessened for our implementation). Regardless of evaluation criterion (maximum, mean or median running time) the `BKN` algorithm dominates in performance[3].

[3] Due to technical constraints, there is system overhead negatively influencing the measured time for `BKN`. This is particularly noticeable for instances of the `berlin` data set since they have so few vertices. Their approach is the fastest overall in our measurements, but it is arguably still faster.

Table 3 Summary of running times in seconds. For both `Unleashed` (`accept_approx = false`) and `Exact_Sweeline`. The best times across the two methods are in blue.

Data set	Max	Mean	StdDev	Median	80th %ile	90th %ile	95th %ile	99th %ile
				`Unleashed (accept_approx=false)`				
geolife	233.6160	0.3673	3.7389	**0.0069**	**0.0621**	0.3263	1.0108	6.4489
pigeons	>300	0.8508	9.0841	**0.0222**	**0.1645**	0.9395	3.7772	13.5008
athens	29.1741	0.2133	1.3476	**0.0190**	**0.1091**	**0.2808**	**0.5125**	3.6473
OV	**24.6879**	**1.2132**	3.2387	**0.0580**	**1.1886**	**3.7313**	**5.2031**	**15.0121**
chicago	0.4679	**0.0036**	0.0147	**0.0024**	**0.0040**	**0.0053**	**0.0063**	**0.0088**
berlin	0.0059	**0.0001**	0.0001	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0002**
uniD	**0.1402**	**0.0023**	0.0048	**0.0019**	**0.0027**	**0.0032**	**0.0041**	**0.0076**
drifter	61.2818	0.1199	1.1781	**0.0091**	**0.0555**	0.1919	0.4474	1.9376
				`Exact (with_dijkstra=false)`				
geolife	**17.5543**	**0.0984**	0.1903	0.0505	0.1330	**0.2312**	**0.3468**	**0.7444**
pigeons	**2.9708**	**0.1649**	0.1472	0.1265	0.2305	**0.3153**	**0.4130**	**0.7262**
athens	**2.6424**	**0.1617**	0.2359	0.0851	0.2136	0.3816	0.5906	**1.2356**
OV	28.2322	4.5376	6.0343	1.5410	8.5852	13.5812	16.9130	23.8446
chicago	**0.0522**	0.0075	0.0073	0.0043	0.0124	0.0179	0.0231	0.0326
berlin	**0.0029**	0.0004	0.0002	0.0003	0.0004	0.0005	0.0007	0.0014
uniD	0.2809	0.0077	0.0102	0.0064	0.0096	0.0114	0.0139	0.0266
drifter	**1.6604**	**0.0743**	0.0889	0.0505	0.1123	**0.1609**	**0.2215**	**0.4069**

We illustrate the performance profile of these algorithms using Dolan–Moré plots (Figures 9–12). These plots can be read as follows. The x-axis shows a factor $\tau \geq 1$, indicating how many times slower an algorithm may be compared to the best on each instance. The y-axis shows the percentage of instances on which this holds. Thus, if an algorithm's plot contains the point (p, τ) then on p percent of the instances, it is τ times slower than the best performing algorithm. Similarly, $\tau = 1$ reveals how often each algorithm is the best, while a horizontal line such as $p = 90$ indicates how much slower the worst 10% of cases become (which, for `BKN`, is minimal). As τ increases, the curves show how quickly each algorithm "catches up". An algorithm is considered better if its curve approaches the horizontal line $p = 100$ more rapidly. These profiles mirror our observations: `BKN` is consistent; our method lags on median instances, however if the data set contains high enough complexity curves, it frequently and significantly overtakes `Unleashed` as we approach the worst case.

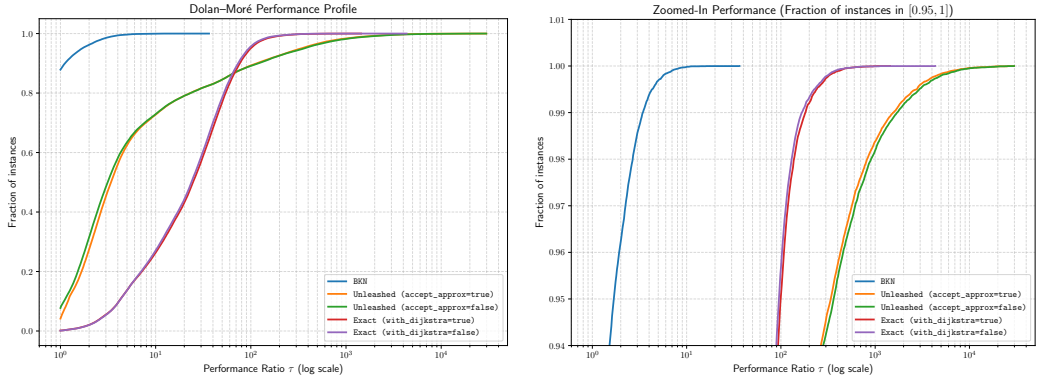


Figure 9 Dolan–Moré performance profile on all pairs of the first 300 curves from the `geolife` data set (44850 in total) from [26].

Table 4 Running time statistics (s). The best running time in bold, the second best in blue.

Data set	Max	Mean	StdDev	Median	80th %ile	90th %ile	95th %ile	99th %ile
				BKN				
geolife	**0.0767**	**0.0024**	0.0016	**0.0021**	**0.0027**	**0.0034**	**0.0042**	**0.0070**
pigeons	**0.1143**	**0.0040**	0.0022	**0.0035**	**0.0048**	**0.0066**	**0.0076**	**0.0102**
athens	**0.0075**	**0.0027**	0.0008	**0.0025**	**0.0033**	**0.0038**	**0.0043**	**0.0055**
OV	**0.0728**	**0.0208**	0.0206	**0.0111**	**0.0376**	**0.0580**	**0.0629**	**0.0682**
chicago	**0.0047**	**0.0018**	0.0002	**0.0018**	**0.0019**	**0.0020**	**0.0021**	**0.0024**
berlin	**0.0032**	0.0016	0.0002	0.0015	0.0017	0.0018	0.0018	0.0019
uniD	**0.0046**	**0.0018**	0.0002	**0.0018**	**0.0020**	**0.0021**	**0.0022**	**0.0024**
drifter	**0.0144**	**0.0024**	0.0009	**0.0022**	**0.0027**	**0.0032**	**0.0037**	**0.0060**
				Unleashed (accept_approx=true)				
geolife	250.5627	0.3367	3.5109	0.0074	0.0623	0.3244	0.9207	5.8376
pigeons	274.5574	0.7529	4.6066	0.0225	**0.1623**	0.9021	3.7495	13.2745
athens	29.4879	0.1754	1.1123	0.0195	**0.1081**	0.2415	0.4817	2.4792
OV	24.8995	1.2463	3.2985	0.0609	1.3038	3.8347	5.5689	15.1980
chicago	0.5972	**0.0035**	0.0144	**0.0023**	**0.0040**	0.0054	0.0064	0.0093
berlin	0.1214	**0.0001**	0.0018	**0.0001**	**0.0001**	**0.0001**	**0.0002**	**0.0002**
uniD	**0.1239**	0.0024	0.0045	0.0020	0.0029	0.0035	0.0042	0.0086
drifter	60.7184	0.1177	1.1667	0.0095	**0.0529**	0.1730	0.4747	1.8644
				Unleashed (accept_approx=false)				
geolife	233.6160	0.3673	3.7389	**0.0069**	**0.0621**	0.3263	1.0108	6.4489
pigeons	>300	0.8508	9.0841	**0.0222**	0.1645	0.9395	3.7772	13.5008
athens	29.1741	0.2133	1.3476	**0.0190**	0.1091	0.2808	0.5125	3.6473
OV	**24.6879**	**1.2132**	3.2387	**0.0580**	**1.1886**	**3.7313**	**5.2031**	**15.0121**
chicago	0.4679	0.0036	0.0147	0.0024	0.0040	**0.0053**	**0.0063**	**0.0088**
berlin	0.0059	**0.0001**	0.0001	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0002**
uniD	0.1402	**0.0023**	0.0048	**0.0019**	**0.0027**	**0.0032**	**0.0041**	**0.0076**
drifter	61.2818	0.1199	1.1781	**0.0091**	0.0555	0.1919	0.4474	1.9376
				Exact (with_dijkstra=true)				
geolife	**7.6259**	0.1028	0.1874	0.0525	0.1381	0.2414	0.3633	0.7742
pigeons	3.1337	0.1672	0.1563	0.1268	0.2322	0.3201	0.4239	0.7432
athens	5.3387	0.1975	0.3426	0.0912	0.2434	0.4529	0.7634	1.6910
OV	24.4039	3.3321	5.0168	0.7700	6.7386	10.2982	14.0325	19.9612
chicago	0.0543	0.0076	0.0076	0.0043	0.0122	0.0182	0.0241	0.0344
berlin	0.0037	**0.0004**	0.0002	**0.0003**	**0.0004**	**0.0004**	0.0006	**0.0014**
uniD	0.2445	0.0077	0.0098	0.0064	0.0095	0.0114	0.0138	0.0271
drifter	1.7855	0.0806	0.1061	0.0519	0.1182	0.1748	0.2428	0.4936
				Exact (with_dijkstra=false)				
geolife	17.5543	**0.0984**	0.1903	0.0505	0.1330	**0.2312**	**0.3468**	**0.7444**
pigeons	**2.9708**	**0.1649**	0.1472	0.1265	0.2305	**0.3153**	**0.4130**	**0.7262**
athens	**2.6424**	**0.1617**	0.2359	0.0851	0.2136	**0.3816**	**0.5906**	**1.2356**
OV	28.2322	4.5376	6.0343	1.5410	8.5852	13.5812	16.9130	23.8446
chicago	**0.0522**	0.0075	0.0073	0.0043	0.0124	0.0179	0.0231	0.0326
berlin	**0.0029**	**0.0004**	0.0002	**0.0003**	**0.0004**	0.0005	0.0007	**0.0014**
uniD	0.2809	0.0077	0.0102	0.0064	0.0096	0.0114	0.0139	0.0266
drifter	**1.6604**	**0.0743**	0.0889	0.0505	0.1123	**0.1609**	**0.2215**	**0.4069**

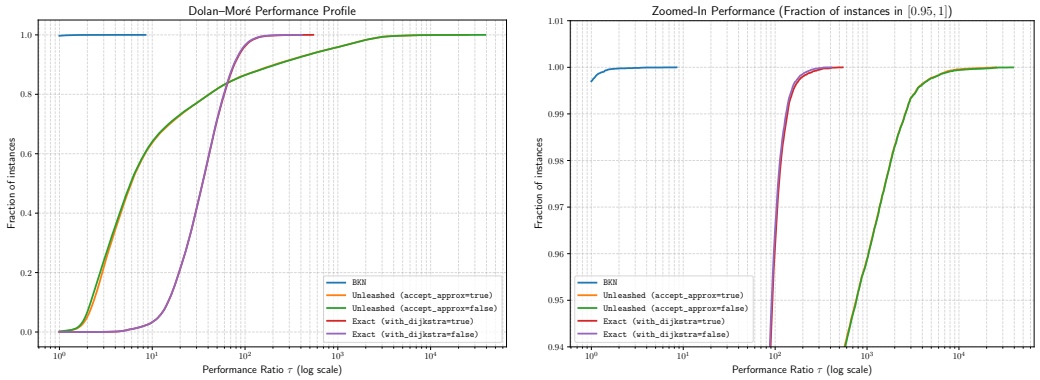


Figure 10 Dolan–Moré performance profile on all pairs of the first 300 curves from the `pigeons` data set (44850 in total) from [26].

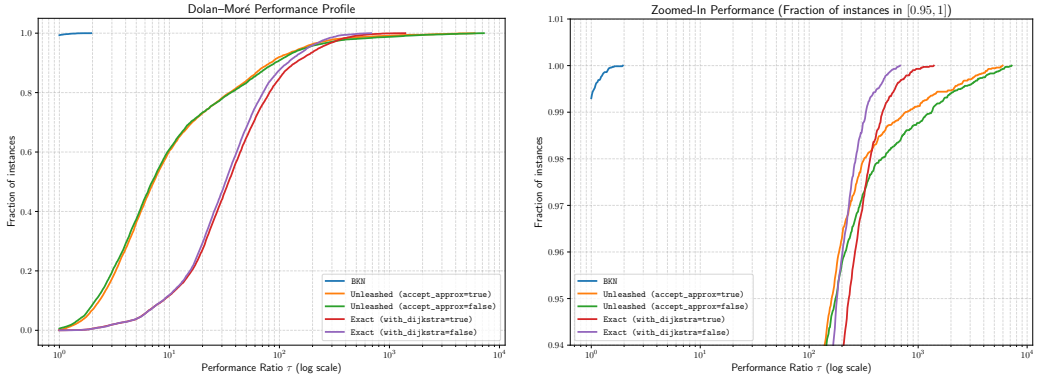


Figure 11 Dolan–Moré profile on pairs from the `athens` data set (7139 in total) from [35].

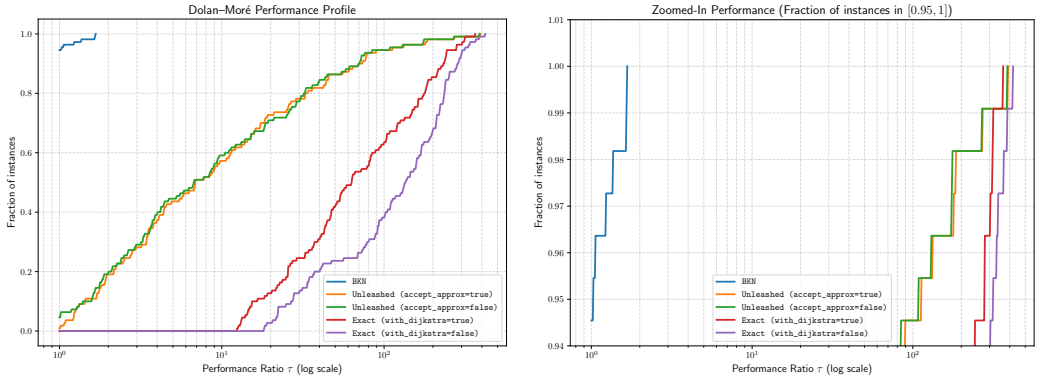


Figure 12 Dolan–Moré performance profile on all pairs from the `OV` data set (110 in total).

6 Conclusion

We studied the frog-based framework for computing the Fréchet distance introduced in [26]. On the theoretical side, we identified three aspects of the original algorithm that incur avoidable inaccuracies in the distance computation or introduce avoidable heuristic overhead.

Specifically, the method of [26] consists of two main components. The first is the computation of the Fréchet distance between (π, σ) via a discrete weighted graph whose complexity is increased iteratively each round, through a process they call *refinement*. Their process heuristically bisects edges of (π, σ), increasing the graph complexity, until the discrete min-cost path through this graph realises the Fréchet distance. We provide a new refinement scheme that offers worst-case guarantees on this process. Our more elaborate refinement is careful not to drastically increase the time per round, yet it yields an explicit upper bound on the convergence rate – which we consider theoretically meaningful.

The second component is a lossless curve simplification procedure that produces curves $(\overline{\pi}, \overline{\sigma})$ of lower complexity such that $\mathcal{D}_F(\pi, \sigma) = \mathcal{D}_F(\overline{\pi}, \overline{\sigma})$ – effectively reducing the input size. Although correct, the original simplification is overly conservative: it relies on a lower bound that is unnecessarily weak, and it avoids geometric corner cases by aggressively propagating vertex slack across a constant but wide neighbourhood. Exploiting the weighted-graph structure already inherent in the frog-based method, we present a more refined construction that attains a higher lower bound. Consequently, our simplifications have lower complexity.

Our primary contribution is an empirical evaluation. We supply an openly available test suite, which enables a reproducible comparison between frog-based implementations and the highly optimised algorithm of Bringmann, Künnemann and Nusser [10]. Our results stand in contrast to the conclusions of [26]: On our broader range of real-world data sets, `BKN` consistently outperforms both frog-based variants. Notably, after removing near-duplicate consecutive points through simplification, the data set curve have *very* few vertices remaining for modern hardware. We have reason to believe that, on larger inputs, the performance gap between our method and `Unleashed` widens. However, we are not aware of real-world benchmark sets whose median complexity exceeds 1,000 "distinct" vertices, and synthetic data behaves entirely differently under lossless simplification. We argue that existing benchmarks are becoming outdated and that the community should develop more modern ones.

We note that the `BKN` algorithm is currently very fast, but still inexact (although it obtains very good approximation ratios). We find it an interesting open problem to augment this code base to allow for exact computations. We note that this is not as straightforward as adding an exact computation kernel (which, the codebase variant in CGAL already has) but that it also requires algorithmic changes to change its decision variant algorithm into the actual computation of the Fréchet distance.

Regarding the experimental results, our findings might seem discouraging: our implementation and analysis describe an approach that does not, in practice, match the performance of the current state-of-the-art. However, we consider this outcome important for the community. Our study offers a transparent and verifiable assessment of the frog-based method and demonstrates that its practical behaviour is volatile and underwhelming.

At the same time, our analysis reaffirms that the frog-based framework is theoretically compelling. We show that this method in particular enables a significantly more fine-grained lossless simplification procedure than general-purpose methods. Nevertheless, our experiments indicate that its main value lies in its theoretical insights rather than in practical performance.

References

1 Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 898–917, USA, 2018. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611975031.58.

2 Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the Discrete Fréchet Distance in Subquadratic Time. *SIAM Journal on Computing*, 43(2):429–449, 2014. doi:10.1137/130920526.

3 Pankaj K Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. Subtrajectory clustering: Models and algorithms. *ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pages 75–87, 2018. doi:10.1145/3196959.3196972.

4 Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, and Carola Wenk. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica*, 19:601–632, 2015. doi:10.1007/s10707-014-0222-6.

5 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995. doi:10.1142/S0218195995000064.

6 Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet Distance for Curves, Revisited. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA)*, volume 4168 of *Lecture Notes in Computer Science*, pages 52–63. Springer, 2006. doi:10.1007/11841036_8.

7 Lotte Blank, Jacobus Conradi, Anne Driemel, Benedikt Kolbe, André Nusser, and Marena Richter. Transforming Dogs on the Line: On the Fréchet Distance Under Translation or Scaling in 1D. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:16, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2025.22.

8 Alessandro Bombelli, Lluis Soler, Eric Trumbauer, and Kenneth D Mease. Strategic Air Traffic Planning with Fréchet distance aggregation and rerouting. *Journal of Guidance, Control, and Dynamics*, 40(5):1117–1129, 2017. doi:10.2514/1.G002308.

9 Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *International Conference on Very Large Data Bases (VLDB)*, pages 853–864, 2005. doi:10.5555/1083592.1083691.

10 Karl Bringmann, Marvin Künnemann, and André Nusser. Walking the Dog Fast in Practice: Algorithm Engineering of the Fréchet Distance. In *Symposium on Computational Geometry (SoCG)*, volume 129 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:21, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SOCG.2019.17.

11 Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristan, Rodrigo I Silveira, Frank Staals, and Carola Wenk. Clustering trajectories for map construction. *ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 1–10, 2017. doi:10.1145/3139958.3139964.

12 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Jorren Hendriks, Erfan Hosseini Sereshgi, Vera Sacristán, Rodrigo I Silveira, Jorrick Sleijster, Frank Staals, and Carola Wenk. Improved map construction using subtrajectory clustering. *ACM Workshop on Location-Based Recommendations, Geosocial Networks, and Geoadvertising (SIGSPATIAL)*, pages 1–4, 2020. doi:10.1145/3423334.3431451.

13 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry & Applications*, 21(03):253–282, 2011. doi:10.1142/S0218195911003638.

14 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. *Four Soviets Walk the Dog—with an Application to Alt's Conjecture*, pages 1399–1413. SIAM, 2014. doi:10.1137/1.9781611973402.103.

15 Maike Buchin, Bernhard Kilgus, and Andrea Kölzsch. Group diagrams for representing trajectories. *International Journal of Geographical Information Science*, 34(12):2401–2433, 2020. doi:10.1145/3283207.3283208.

16 Jacobus Conradi and Anne Driemel. Finding complex patterns in trajectory data via geometric set cover. *arXiv preprint arXiv:2308.14865*, 2023. doi:10.48550/arXiv.2308.14865.

17 Jacobus Conradi and Anne Driemel. On Computing the k-Shortcut Fréchet Distance. *ACM Trans. Algorithms*, 20(4), August 2024. doi:10.1145/3663762.

18 Jacobus Conradi, Anne Driemel, and Benedikt Kolbe. Revisiting the Fréchet distance between piecewise smooth curves. *Computational Geometry*, 129:102194, 2025. doi:10.1016/j.comgeo.2025.102194.

19 Jacobus Conradi, Benedikt Kolbe, Ioannis Psarros, and Dennis Rohde. Fast approximations and coresets for (k,ℓ)-median under dynamic time warping. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry, SoCG 2024, Athens, Greece, June 11-14, 2024*, volume 293 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SOCG.2024.42.

20 Jacobus Conradi, Ivor van der Hoog, and Eva Rotenberg. On computing the (exact) Fréchet distance with a frog, 2025. arXiv:2512.07728.

21 Thomas Devogele. A new merging process for data integration based on the discrete Fréchet distance. In *Advances in spatial data handling*, pages 167–181, 2002. doi:10.1007/978-3-642-56094-1_13.

22 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. In *International Symposium on Computational Geometry (SoCG)*, pages 365–374. ACM, 2010. doi:10.1145/1810959.1811019.

23 Anne Driemel, Ivor van der Hoog, and Eva Rotenberg. On the Discrete Fréchet Distance in a Graph. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2022.36.

24 University of Technology Eindhoven. MoveTK: the movement toolkit. https://github.com/movetk/movetk, 2020.

25 Sariel Har-Peled and Benjamin Raichel. The Fréchet distance revisited and extended. In *Symposium on Computational Geometry (SoCG)*, pages 448–457, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1998196.1998269.

26 Sariel Har-Peled, Benjamin Raichel, and Eliot W. Robson. The Fréchet Distance Unleashed: Approximating a Dog with a Frog. In Oswin Aichholzer and Haitao Wang, editors, *International Symposium on Computational Geometry (SoCG)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:13, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2025.54.

27 Jincai Huang, Min Deng, Jianbo Tang, Shuling Hu, Huimin Liu, Sembeto Wariyo, and Jinqiang He. Automatic generation of road maps from low quality GPS trajectory data via structure learning. *IEEE Access*, 6:71965–71975, 2018. doi:10.1109/ACCESS.2018.2882581.

28 Minghui Jiang, Ying Xu, and Binhai Zhu. Protein structure–structure alignment with discrete Fréchet distance. *Journal of bioinformatics and computational biology*, 6(01):51–64, 2008. doi:10.1142/s0219720008003278.

29 Maximilian Konzack, Thomas McKetterick, Tim Ophelders, Maike Buchin, Luca Giuggioli, Jed Long, Trisalyn Nelson, Michel A Westenberg, and Kevin Buchin. Visual analytics of delays and interaction in movement data. *International Journal of Geographical Information Science*, 31(2):320–345, 2017. doi:10.5555/3048386.3048392.

30 Ariane Mascret, Thomas Devogele, Iwan Le Berre, and Alain Hénaff. Coastline matching process based on the discrete Fréchet distance. In *Progress in Spatial Data Handling*, pages 383–400. Springer, 2006. doi:10.1007/3-540-35589-8_25.

31 Tobias Moers, Lennart Vater, Robert Krajewski, Julian Bock, Adrian Zlocki, and Lutz Eckstein. The exiD Dataset: A Real-World Trajectory Dataset of Highly Interactive Highway Scenarios in Germany. *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 958–964, 2022. doi:10.1109/IV51971.2022.9827305.

32 Roniel Sousa, Azzedine Boukerche, and Antonio Loureiro. Vehicle Trajectory Similarity: Models, Methods, and Applications. *ACM Computing Surveys*, 53(5), 2020. doi:10.1145/3406096.

33 E Sriraghavendra, K Karthik, and Chiranjib Bhattacharyya. Fréchet distance based approach for searching online handwritten documents. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, pages 461–465, 2007. doi:10.1109/ICDAR.2007.4378752.

34 Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. A survey of trajectory distance measures and performance evaluation. *Proceedings of the VLDB Endowment*, 29(1):3–32, 2020. doi:10.1007/s00778-019-00574-9.

35 Ivor van der Hoog, Lara Ost, Eva Rotenberg, and Daniel Rutschmann. Efficient Greedy Discrete Subtrajectory Clustering. In *International Symposium on Computational Geometry (SoCG)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 78:1–78:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SOCG.2025.78.

36 Suyi Wang, Yusu Wang, and Yanjie Li. Efficient map reconstruction and augmentation via topological methods. *ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 1–10, 2015. doi:10.1145/2820783.2820833.

37 Dong Xie, Feifei Li, and Jeff M Phillips. Distributed trajectory similarity search. *Proceedings of the VLDB Endowment*, 10(11):1478–1489, 2017. doi:10.14778/3137628.3137655.

Upward Book Embeddings of Partitioned Digraphs

Giordano Da Lozzo
ICITA Department, Roma Tre University, Italy

Fabrizio Frati
ICITA Department, Roma Tre University, Italy

Ignaz Rutter
Faculty of Computer Science and Mathematics, University of Passau, Germany

Abstract

In 1999, Heath, Pemmaraju, and Trenk [SIAM J. Comput. 28(4), 1999] extended the classic notion of book embeddings to digraphs, introducing the concept of *upward book embeddings*, in which the vertices must appear along the spine in a topological order and the edges are partitioned into pages, so that no two edges in the same page cross. For a partitioned digraph $G = (V, \bigcup_{i=1}^{k} E_i)$, that is, a digraph whose edge set is partitioned into k subsets, an upward book embedding is required to assign edges to pages as prescribed by the given partition. In a companion paper, Heath and Pemmaraju [SIAM J. Comput. 28(5), 1999] proved that the problem of testing the existence of an upward book embedding of a partitioned digraph is linear-time solvable for $k = 1$ and recently Akitaya, Demaine, Hesterberg, and Liu [GD, 2017] have shown the problem NP-complete for $k \geq 3$. In this paper, we study upward book embeddings of partitioned digraphs and focus on the unsolved case $k = 2$. Our first main result is a novel characterization of the *upward embeddings* that support an upward book embedding in two pages. We exploit this characterization in several ways, and obtain a rich picture of the complexity landscape of the problem. First, we show that the problem remains NP-complete when $k = 2$, thus closing the complexity gap for the problem. Second, we show that, for an n-vertex partitioned digraph with a prescribed planar embedding, the existence of an upward book embedding that respects the given planar embedding can be tested in $O(n \log^3 n)$ time. Finally, leveraging the SPQ(R)-tree decomposition of biconnected graphs into triconnected components, we present a cubic-time testing algorithm for biconnected directed partial 2-trees.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Graph algorithms; Theory of computation → Design and analysis of algorithms

Keywords and phrases upward book embeddings, partitioned digraphs, SPQ-trees, 2-trees

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.36

Related Version *Full Version*: `https://arxiv.org/abs/2603.17128` [21]

Acknowledgements This research was initiated at the Fourth Summer Workshop on Graph Drawing (SWGD 2024). Thanks to the organizers and the participants for an inspiring atmosphere.

1 Introduction

Book embeddings are a classic and influential topic in combinatorial and algorithmic graph theory. The notion of a book as a topological space was introduced in the late 60s by Persinger [48] and Atneosen [5], and later developed in its current and more popular form by the seminal work of Ollmann [47]. In a ***book embedding*** of a graph $G = (V, E)$, all vertices lie along a line – referred to as the ***spine*** – while edges are placed into distinct half-planes bounded by the spine, known as the ***pages*** of the book. Therefore, constructing such an embedding for G amounts to computing a pair (π, σ), where $\pi : V \leftrightarrow \{1, \ldots, |V|\}$ is a linear ordering of the vertices and $\sigma : E \to \{1, \ldots, k\}$ is a partition of the edges into k pages so that no two edges in the same page cross according to π, i.e., their end-vertices do not alternate in π; see Fig. 1a for an example. The minimum value of k for which this is possible is the ***book thickness*** of G (also called ***stack number*** or ***page number***).

© Giordano Da Lozzo, Fabrizio Frati, and Ignaz Rutter;
licensed under Creative Commons License CC-BY 4.0

Research on book embeddings originated from problems in VLSI circuit design [20], and has since found applications in sorting permutations [49, 52], fault-tolerant processing [50], compact graph encodings [40, 46], graph drawing [14, 25, 29, 33], computational origami [1, 45], parallel process scheduling [13], parallel matrix computations [36], and other domains. For a more comprehensive overview of applications, see [28]. The notion of book embedding was extended to digraphs by Heath, Pemmaraju, and Trenk [37] by introducing the natural requirement that in a book embedding (π, σ) of a digraph G the ordering π must be a topological ordering of G; see Figs. 1b and 1c for an example. Such book embeddings are called ***upward book embeddings*** as they are naturally depicted with vertices placed on a vertical line and edges drawn as arcs monotonically increasing in the y-direction in their page.

Next, we provide an overview of the major algorithmic results on book embeddings; in the full version [21], we also discuss combinatorial results. Concerning undirected graphs, already in 1979, Bernhart and Kainen [10] showed that the graphs of book thickness 1 are the outerplanar graphs and that the graphs of book thickness 2 are the sub-Hamiltonian (planar) graphs. Whereas the former are recognizable in linear time [54], recognizing the latter is NP-complete, even for planar triangulations [55]. Recently, Ganian et al. [31] presented a sub-exponential-time algorithm for testing sub-Hamiltonicity, which is asymptotically tight under ETH. Concerning digraphs, the problem of testing the existence of an upward book embedding in k pages is called Upward Book Embedding. For more than two decades, the only known NP-completeness result for the problem was the one shown by Heath and Pemmaraju [35] when $k = 6$. Recently, in two subsequent papers, Binucci et al. [16] and Bekos et al. [8, 9] closed the computational gap by showing NP-completeness for $k \geq 3$ and $k = 2$, respectively. These results, together with the linear-time algorithm for $k = 1$ [35], completely characterize the complexity of the Upward Book Embedding problem with respect to k. For $k = 2$, efficient algorithms have been devised for special classes [15, 42, 43].

Partitioned book embeddings. Due to the NP-hardness of testing the existence of a book embedding (π, σ), it is natural to study the complexity of the problem if π or σ is given as part of the input. If the vertex ordering π is fixed, the problem becomes the k-coloring problem for circle graphs. The problem is clearly linear-time solvable for $k \leq 2$. Unger [53] showed that it is NP-complete for $k \geq 4$. The complexity of the case $k = 3$ is still open [6, 7], although a

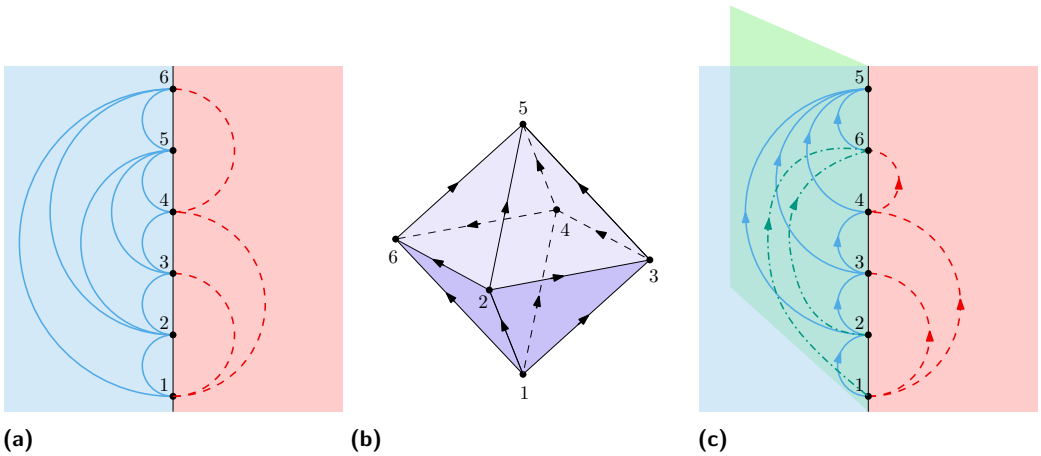


Figure 1 (a) A book embedding of the octahedron in 2 pages. (b) An orientation of the octahedron. (c) An upward book embedding of the directed octahedron in (b) in 3 pages, which is optimal.

Table 1 Complexity of the upward book embedding problem. Results marked ✓ are trivial.

		vertex order π	
		fixed	**variable**
page assignment σ	**fixed**	O(n+m) time ✓	1 page: O(n) time [35] 2 pages: NP-complete (Theorem 4) $\geq$ 3 pages: NP-complete [1]
	variable	2 pages: O(n) time ✓ 3 pages: OPEN [7] $\geq$ 4 pages: NP-complete [53]	2 pages: NP-complete [9] $\geq$ 3 pages: NP-complete [16]

quasi-polynomial-time algorithm for $k = 3$ has recently been proposed [51]. Conversely, if the partition σ into k pages is given, for undirected graphs, there are polynomial-time algorithms for $k = 1$ [35] and $k = 2$ [4, 38, 39], whereas the problem is NP-complete for $k \geq 3$ [3]. For directed graphs, in which the vertex ordering has to be a topological ordering, the problem is called Partitioned Upward Book Embedding. For $k = 1$ it coincides with the "unpartitioned" case, solved in [35, 37]. For $k > 1$, it was first studied by Akitaya et al. [1]. They connected the problem to applications in map folding [45] and attributed it to Edmonds, who, in 1997, posed the question specifically for $k = 4$ when the edges assigned to each page form a matching. They showed NP-completeness for $k = 3$ and for $k \geq 4$ even if the edges in each page form a matching. For $k = 2$, they gave a linear-time algorithm for the special case when the edges in each page form a matching. However, they left the complexity of the general case for $k = 2$ open. See Table 1 for an overview of the complexity landscape.

Our Contributions. We study the Partitioned Upward Book Embedding problem in the unsolved case $k = 2$. As every upward book embedding corresponds to an upward planar drawing, i.e., the edges are drawn as non-crossing y-monotone curves, we make use of combinatorial descriptions of such drawings, called ***upward embeddings*** [11]. Our first main result is a characterization of the upward embeddings that support an upward book embedding in two pages. We exploit this characterization in several ways and obtain a rich picture of the complexity landscape of the problem. First, we show that the problem remains NP-complete when $k = 2$, thus closing the complexity gap and exhibiting a sharp contrast with the undirected case, where the problem is linear-time solvable [39]. Our proof also implies that the problem is W[1]-hard with respect to the treewidth. Second, we show that, when the planar embedding is prescribed, the problem can be solved in $O(n \log^3 n)$ time. Our algorithm is inspired by the network-flow approach of Bertolazzi et al. [11] and requires the use of several non-trivial ingredients arising from our characterization. Finally, in the variable-embedding setting, we present a cubic-time algorithm for biconnected directed partial 2-trees. Our algorithm exploits a compact representation, called *descriptor pair*, of the features of upward embeddings of subgraphs that admit an extension to an upward embedding, satisfying our characterization, of the entire graph. This allows us to compute, for the subgraph associated with each node of the SPQ(R)-tree, its set of realizable descriptor pairs by dynamic programming. While we focus on the decision problem, all our algorithms are constructive and can be modified to compute an upward book embedding, if one exists. A full version of the paper with complete proofs can be found in [21].

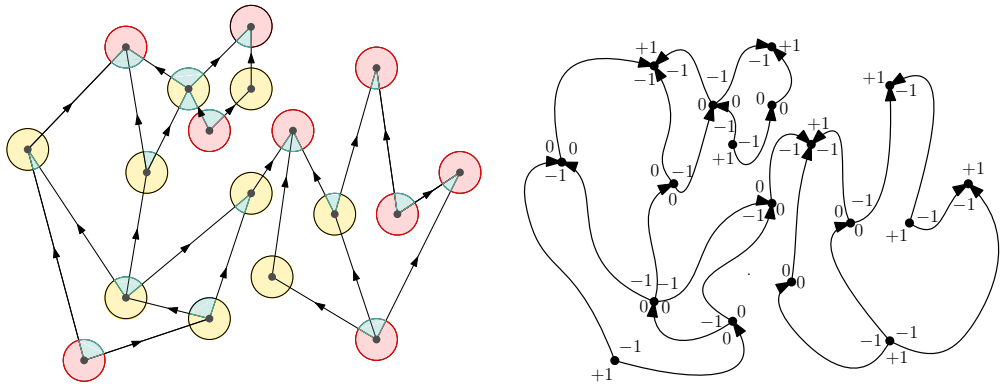


Figure 2 (a) An upward planar drawing Γ and its big, small, and flat angles depicted as red, green, and yellow sectors, respectively. (b) The upward-consistent angle assignment λ_Γ defined by Γ.

2 Preliminaries

For basic definitions on graphs and their drawings see [22, 27] or the full version of the paper.

Two planar drawings of a connected digraph are ***topologically equivalent*** if they have the same clockwise order of the edges around each vertex and the same clockwise order of the edges along the boundary of the outer face. A ***planar embedding*** is a class of topologically equivalent planar drawings. A ***plane digraph*** is a planar digraph together with a planar embedding. A planar embedding of a digraph is ***bimodal*** if, for every vertex v, the edges that have their tail at v are consecutive in the clockwise order of the edges incident to v.

Upward planarity. A ***switch*** in a digraph G is a source or a sink. A drawing of G is ***upward*** if every edge is represented by a Jordan arc that is strictly increasing in the y-direction from its tail to its head. If G admits an upward planar drawing, it is an ***upward planar digraph***. Let $\mathcal{E}$ be a bimodal planar embedding of G. An ***angle*** at a vertex v is an ordered pair (e_1, e_2) of edges incident to v where e_2 immediately follows e_1 in clockwise order around v. An angle (e_1, e_2) is a ***switch angle*** if v is the head or the tail of both e_1 and e_2, otherwise it is a ***flat angle***. The set of angles of $\mathcal{E}$ is denoted as $\mathcal{A}_\mathcal{E}$. An ***angle assignment for $\mathcal{E}$*** is a function $\lambda\colon \mathcal{A}_\mathcal{E} \to \{-1, 0, 1\}$, and it is ***upward-consistent*** if it satisfies the following conditions: **(C1)** For each angle a in $\mathcal{A}_\mathcal{E}$, we have $\lambda(a) = 0$ if and only if a is flat. **(C2)** For each vertex v of G, we have $\sum_a \lambda(a) = 2 - \deg(v)$, where the sum is over the angles at v. **(C3)** For each face f of $\mathcal{E}$, we have $\sum_a \lambda(a) = -2$ if f is an internal face and $\sum_a \lambda(a) = 2$ if f is the outer face, where the sum is over the angles in f.

Let Γ be an upward planar drawing with planar embedding $\mathcal{E}$, see Fig. 2. Clearly, $\mathcal{E}$ is bimodal. Also, it defines an angle assignment λ_Γ as follows. Let $a = (e_1, e_2)$ be an angle of $\mathcal{E}$. If a is flat, then $\lambda_\Gamma(a) = 0$. Otherwise, consider the geometric angle α in Γ corresponding to a. We define $\lambda_\Gamma(a) = -1$ if $\alpha < \pi$ and $\lambda_\Gamma(a) = 1$ if $\alpha > \pi$. Note that λ_Γ is upward-consistent.

A pair $(\mathcal{E}, \lambda)$ of a planar embedding $\mathcal{E}$ and an angle assignment λ is an ***upward embedding*** if there exists an upward planar drawing Γ with embedding $\mathcal{E}$ such that $\lambda = \lambda_\Gamma$. Angles a with $\lambda(a) = 0$ are flat; also, we say that an angle a is ***large*** if $\lambda(a) = 1$ and ***small*** if $\lambda(a) = -1$. An angle assignment is determined by *assigning* each switch to an incident angle, which corresponds to making that angle large; a switch angle to which no switch is assigned is small. A digraph equipped with an upward embedding is an ***upward plane digraph***.

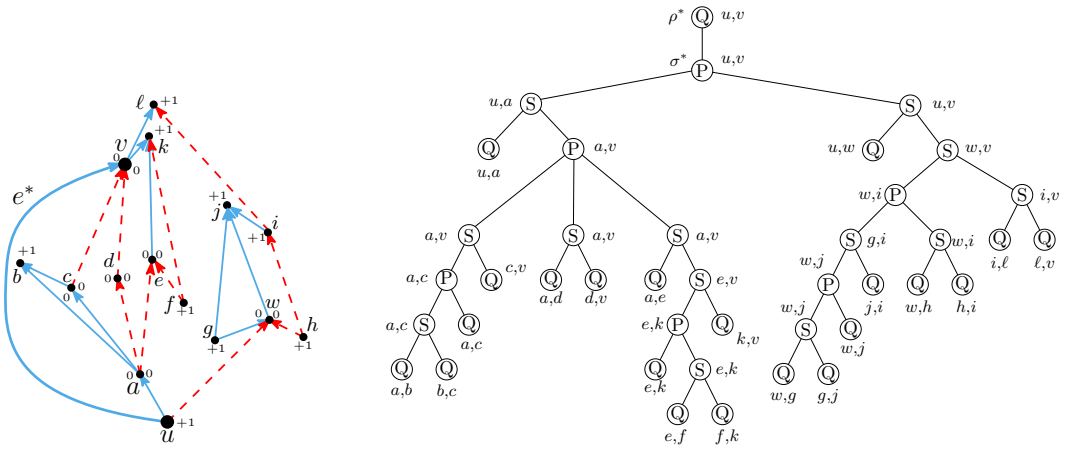


Figure 3 (a) An upward planar drawing Γ of a biconnected partitioned directed partial 2-tree G. The labeling λ_Γ of the angles determined by Γ is shown; the missing labels are equal to -1. (b) The SPQ-tree of G rooted at the Q-node corresponding to the edge e^* of G.

▶ **Theorem 1** ([11, 26]). *Let $\mathcal{E}$ be a planar embedding and λ be an angle assignment for $\mathcal{E}$. The pair $(\mathcal{E}, \lambda)$ is an upward embedding if and only if $\mathcal{E}$ is bimodal and λ is upward-consistent.*

Upward Book Embeddings. An n-vertex ***partitioned digraph*** is a digraph $G = (V, \bigcup_{i=1}^{k} E_i)$ whose edge set is partitioned into k sets. An ***upward book embedding*** (in k pages) of G is a bijection $\pi\colon V \leftrightarrow \{1, \ldots, n\}$ such that: **(i)** for each edge $e = (u, v)$, it holds that $\pi(u) < \pi(v)$, and **(ii)** for any $i \in \{1, \ldots, k\}$, no two edges $(u, v), (w, x) \in E_i$ cross, where (u, v) and (w, x) ***cross*** if $\pi(u) < \pi(w) < \pi(v) < \pi(x)$ or $\pi(w) < \pi(u) < \pi(x) < \pi(v)$. We focus on the case $k = 2$ and let $L := E_1$ and $R := E_2$. We often omit that our upward book embeddings are in two pages and call ***left*** and ***right*** the edges in L and R, respectively, since an upward book embedding (π, σ) of $G = (V, L \cup R)$ determines an upward planar drawing of G where the vertices lie along a vertical line, called ***spine***, so that the y-coordinate of each vertex v is $\pi(v)$, and each edge (u, v) in L (resp. in R) is drawn as a semi-circle to the left (resp. to the right) of the spine. In all the illustrations, the edges in L are blue and solid, while the edges in R are red and dashed.

An upward planar drawing ***respects*** a planar embedding $\mathcal{E}$ if it belongs to the equivalence class $\mathcal{E}$. An upward book embedding ***respects*** a planar embedding $\mathcal{E}$ if the associated upward planar drawing Γ does and ***respects*** an upward embedding $(\mathcal{E}, \lambda)$ if it respects $\mathcal{E}$ and $\lambda_\Gamma = \lambda$.

In an upward embedding $(\mathcal{E}, \lambda)$, a vertex v is ***4-modal*** if: **(i)** If v is not a switch, then in clockwise order around v in $\mathcal{E}$ we have all outgoing left edges, all outgoing right edges, all incoming right edges, and all incoming left edges; one of the former two sets and/or one of the latter two sets might be empty. **(ii)** If v is a source (resp. a sink), then in clockwise order around v in $(\mathcal{E}, \lambda)$ we have the large angle at v, all outgoing left edges, and all outgoing right edges (resp. the large angle at v, all incoming right edges, and all incoming left edges); one of the two sets might be empty. Also, $(\mathcal{E}, \lambda)$ is ***4-modal*** if all vertices are 4-modal.

▶ **Property 2.** *Let Γ be an upward book embedding of a partitioned digraph $G = (V, L \cup R)$ and let $(\mathcal{E}, \lambda)$ be the upward embedding of G defined by Γ. Then $(\mathcal{E}, \lambda)$ is 4-modal.*

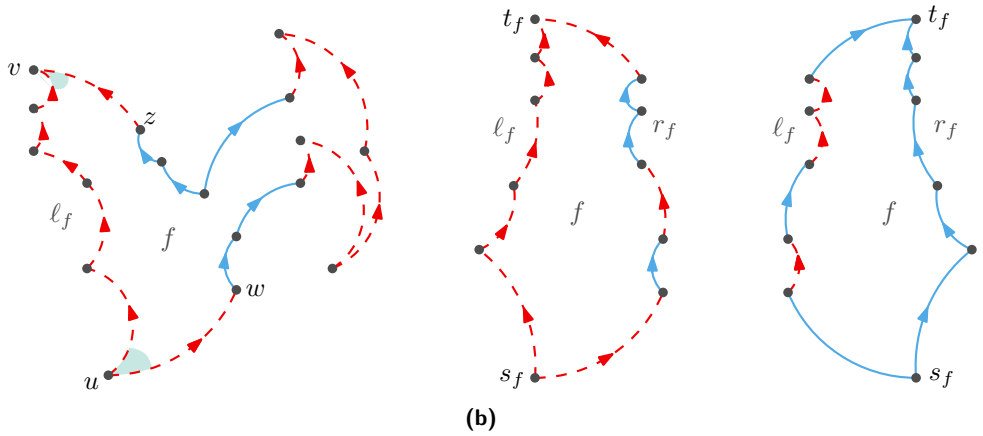


Figure 4 Impossible faces in (a) an upward plane digraph and (b) a plane st-graph.

Partial 2-trees. A graph is an (undirected) ***partial 2-tree*** if, equivalently, it has treewidth at most two, or it excludes K_4 as a minor, or it is a subgraph of a 2-tree, which is a graph that can be obtained from an edge by repeatedly inserting a degree-2 vertex adjacent to two adjacent vertices. Notably, the class of partial 2-trees includes the ***series-parallel graphs***.

Let G be a biconnected partial 2-tree and let $e^* = (u^*, v^*)$ be an edge of G, see Fig. 3. The ***SPQ-tree*** T of G with respect to e^* is a rooted tree that describes a recursive decomposition of G into smaller partial 2-trees; it is a specialization of the well-known ***SPQR-tree***, which is defined for general biconnected planar graphs [24, 34], and can be computed in linear time.

The root of T is a Q-node ρ^* associated with G and with a single child σ^*. Define $G - e^*$ as the ***pertinent graph*** of σ^*, and let u^* and v^* be the ***poles*** of σ^*. We proceed recursively. Suppose we are given a quadruple $\langle \mu, u, v, G_\mu \rangle$, where μ is a node of T with poles u and v, and G_μ is its pertinent graph (initially, this is $\langle \sigma^*, u^*, v^*, G - e^* \rangle$). Three cases can occur:

- If G_μ is a single edge (u, v), then μ is a Q-node representing that edge; μ is a leaf of T.
- If G_μ is not biconnected, then μ is an S-node. Let w be an arbitrary cut-vertex of G_μ. Then μ has two children ν_1 and ν_2 in T corresponding to the two subgraphs G_{ν_1} and G_{ν_2}, respectively, obtained by splitting G_μ at w. The poles of ν_1 are u and w, and those of ν_2 are w and v. The construction of T recurses on $\langle \nu_1, u, w, G_{\nu_1} \rangle$ and on $\langle \nu_2, w, v, G_{\nu_2} \rangle$.
- If G_μ is biconnected, then μ is a P-node. Then μ has k children $\nu_1, \ldots, \nu_k$, where k is the number of subgraphs $G_{\nu_1}, \ldots, G_{\nu_k}$ obtained by splitting G_μ at u and v. For $i = 1, \ldots, k$, the poles of ν_i are u and v, and the construction of T recurses on $\langle \nu_i, u, v, G_{\nu_i} \rangle$.

A ***directed partial 2-tree*** is a digraph whose underlying graph is a partial 2-tree, where the underlying graph is the undirected graph obtained by ignoring the edge directions. An SPQ-tree of a biconnected directed partial 2-tree G is an SPQ-tree of its underlying graph, although the edges of the pertinent graph of each node are oriented as in G.

3 Characterization for Upward Embeddings

In this section we characterize the upward embeddings $(\mathcal{E}, \lambda)$ of a partitioned upward plane digraph $G = (V, L \cup R)$ that allow for the construction of an upward book embedding. For a face f of $\mathcal{E}$, let $\mathcal{L}_f$ (resp. $\mathcal{R}_f$) be the set of maximal directed paths in the boundary of f that consist of edges with f to their right (resp. to their left). The face f is ***impossible*** if it satisfies one of the following two conditions (see Fig. 4a):

(i) $\mathcal{L}_f$ contains a path ℓ_f with the following properties: a) ℓ_f consists of edges in R; b) the rest of the boundary of f is not a single edge; and c) let e_f^u and e_f^v be the edges of ℓ_f incident to the extremes u and v of ℓ_f, respectively; then the angle in f incident to u and to the right of e_f^u and the angle in f incident to v and to the right of e_f^v are small.

(ii) $\mathcal{R}_f$ contains a path r_f with the following properties: a) r_f consists of edges in L; b) the rest of the boundary of f is not a single edge; and c) let e_f^u and e_f^v be the edges of r_f incident to the extremes u and v of r_f, respectively; then the angle in f incident to u and to the left of e_f^u and the angle in f incident to v and to the left of e_f^v are small.

A ***good embedding*** is an upward embedding $(\mathcal{E}, \lambda)$ that is 4-modal and that is such that no face of $\mathcal{E}$ is impossible. We now prove our characterization.

▶ **Theorem 3.** *Let G be a partitioned upward plane digraph with upward embedding $(\mathcal{E}, \lambda)$. Then G admits an upward book embedding respecting $(\mathcal{E}, \lambda)$ if and only if $(\mathcal{E}, \lambda)$ is good.*

Sketch. We sketch the proof for the case where G is connected; the general case is discussed in the full version. For the necessity, let Γ be an upward book embedding of G respecting $(\mathcal{E}, \lambda)$. Then $\mathcal{E}$ is 4-modal by Property 2. Suppose, for a contradiction, that a face f of $\mathcal{E}$ is impossible. W.l.o.g., assume that $\mathcal{L}_f$ contains a path ℓ_f satisfying condition (i) and let u, v, e_f^u, e_f^v be defined as above. Let further w and z be the vertices adjacent to u and v, respectively, such that the edge (u, w) follows e_f^u in clockwise order around u and the edge (z, v) follows e_f^v in counter-clockwise order around v. Since Γ is upward planar, the order of the vertices of ℓ_f along the spine is the same as their order in ℓ_f, with u below v; also, w lies above u and z below v. If w or z lies in the strip S_{uv} delimited by the horizontal lines through u and v, then (u, w) or (z, v) would cross an edge of the path ℓ_f, because the small angles in $(\mathcal{E}, \lambda)$ force the edges (u, w) and (v, z) to be to the right of e_f^u and e_f^v, respectively. However, no point of the spine lies to the right of ℓ_f and in the strip S_{uv}. It follows that w has to lie above v and z below u. This, however, implies that the edges (u, w) and (z, v) cross each other, given that they are both drawn to the right of the spine. This proves the necessity.

For the sufficiency, we build on a procedure by Bertolazzi et al. [12], who showed that any upward plane digraph can be augmented to a plane st-graph by adding edges, while maintaining the upward embedding. A ***plane st-graph*** G is an upward plane digraph with one source s and one sink t that are incident to the outer face. The boundary of each face f of G consists of two directed paths ℓ_f and r_f, called ***left*** and ***right path***, respectively. In the full version, we extend the technique of Bertolazzi et al. and show that, by suitably adding paths and carefully assigning their edges to L and R, we can augment a good embedding of the given partitioned upward plane digraph G to a good embedding of a partitioned plane st-graph. For this reason, we now assume that G is a partitioned plane st-graph. In this case, a face f is impossible if ℓ_f only consists of edges in R and r_f is not a single edge, or if r_f only consists of edges in L and ℓ_f is not a single edge; see Fig. 4b.

To show that a partitioned plane st-graph G with a good embedding $(\mathcal{E}, \lambda)$ admits an upward book embedding Γ, we use the fact that it can be constructed starting from the left path ℓ_{f_O} of the outer face f_O by repeatedly adding the right path of an internal face whose left path already belongs to the graph, see, e.g., [2, 23, 30, 44]. We start by drawing in Γ the edges of ℓ_{f_O} as semi-circles in their assigned side of the spine. When we draw the right path r_f of a face f whose left path ℓ_f already belongs to the subgraph of G drawn in Γ, we distinguish two cases. First, if r_f is a single edge e, then, since G is simple, ℓ_f contains an internal vertex. Since f is not impossible, $e \in R$. Then we just draw e as a semi-circle to

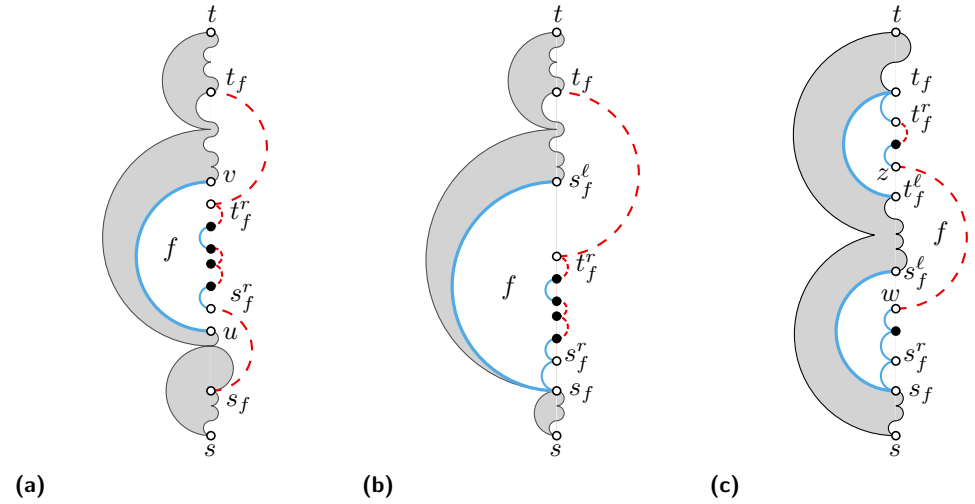


Figure 5 Illustrations for the proof sketch of Theorem 3.

the right of the spine. Second, if r_f is not a single edge, as in Fig. 5, then ℓ_f contains a left edge (u, v), since f is not impossible. If both the first and the last edge of r_f are in R, we place all the internal vertices of r_f on the spine between u and v in the order in which they appear along r_f and draw the edges of r_f as semi-circles in their assigned side of the spine; see Fig. 5a. Otherwise, if the first edge of r_f is in L and its last edge is in R, then also the first edge of ℓ_f is in L due to the 4-modality of $(\mathcal{E}, \lambda)$. We can then choose (u, v) as the first edge of ℓ_f and proceed as discussed in the previous case; see Fig. 5b. The case where the first edge of r_f is in R and its last edge is in L is analogous. Finally, if both the first and the last edge of r_f are in L, then so are the first edge (s_f, s_f^ℓ) and the last edge (t_f^ℓ, t_f) of ℓ_f, by the 4-modality of $(\mathcal{E}, \lambda)$. If ℓ_f is a single edge, then it plays the role of (u, v) and we place the internal vertices of r_f and draw its edges as above. Otherwise, since f is not impossible, r_f contains an edge $(w, z) \in R$. We embed the internal vertices of the subpath of r_f from its start to w, including w, between s_f and s_f^ℓ in the order as they appear along r_f, and we embed the internal vertices of the subpath of r_f from z to its end, including z, between t_f^ℓ and t_f in the order as they appear along r_f, and draw the edges of r_f as semi-circles in their assigned side of the spine; see Fig. 5c. ◀

4 Computational Complexity with Variable Embedding

In this section we study the complexity of testing whether a partitioned digraph $G = (V, L \cup R)$ admits an upward book embedding in two pages. We show that the problem is NP-complete by exploiting Theorem 3 and the NP-hardness of Upward Planarity Testing.

▶ **Theorem 4.** *It is* NP*-complete to decide whether a partitioned planar digraph admits an upward book embedding.*

Proof. Clearly, the problem is in NP. In order to prove NP-hardness, we give a reduction from Upward Planarity Testing, which was proved to be NP-hard by Garg and Tamassia [32]. Given a planar digraph $G = (V, E)$, we construct a partitioned digraph $G' = (V', L \cup R)$ as follows. Subdivide each edge $e \in E$ with a new vertex v_e; these subdivision vertices, together with the vertices in V, form V'. The edge set L consists of the edges outgoing from vertices

in V (and incoming into vertices in $V' \setminus V$), and the edge set R consists of the remaining edges. Clearly, G' can be constructed from G in polynomial time. It remains to show that G' admits an upward book embedding if and only if G admits an upward planar drawing.

For the necessity, observe that an upward book embedding Γ' of G' is an upward planar drawing of G'. Then an upward planar drawing Γ of G is obtained from Γ' by (i) placing each vertex of G in Γ as in Γ' and by (ii) drawing in Γ each edge $e = (u, w)$ of G as the Jordan arc formed by the union of the drawings of (u, v_e) and of (v_e, w) in Γ'.

For the sufficiency, suppose that G admits an upward planar drawing Γ. An upward planar drawing Γ' of G' can be constructed from Γ by placing each subdivision vertex v_e at any internal point of the Jordan arc representing the edge e. Let $(\mathcal{E}', \lambda')$ be the upward embedding corresponding to Γ'. We prove that $(\mathcal{E}', \lambda')$ is 4-modal. Consider any vertex v of G'. If $v \notin V$, then v has one incoming and one outgoing edge in G', hence it is trivially 4-modal. If $v \in V$ then, by construction, all the edges outgoing from v (incoming into v), if any, are in L (resp. are in R) and are consecutive in the clockwise order of the edges incident to v, due to the bimodality of Γ. It follows that v is 4-modal. Finally, $(\mathcal{E}', \lambda')$ has no impossible face, since by construction any maximal directed path in the boundary of any face contains both an edge in L and an edge in R. Hence, $(\mathcal{E}', \lambda')$ is a good embedding. By Theorem 3, we have that G' admits an upward book embedding respecting $(\mathcal{E}', \lambda')$. ◀

Upward Planarity Testing is known to be W[1]-hard with respect to the treewidth [41]. Also, the reduction shown in Theorem 4 constructs a graph which is a subdivision of the original instance of Upward Planarity Testing. Since any two graphs, one of which is a subdivision of the other one, have the same treewidth, we get the following.

▶ **Corollary 5.** *It is W[1]-hard with respect to the treewidth to decide whether a partitioned digraph admits an upward book embedding.*

Furthermore, by subdividing the edges twice, rather than once, so that the edges incident to vertices in V are in L and the other edges in R, the reduction gives an instance where one edge part induces a matching and the other one a forest of stars. This is in sharp contrast with the fact that the problem is linear-time solvable when both edge parts are matchings [1].

5 Test for Graphs with a Fixed Planar Embedding

In this section we show how to exploit the characterization of Theorem 3 in order to prove that, for an n-vertex partitioned plane digraph G with a given planar embedding $\mathcal{E}$, it can be tested in $O(n \log^3 n)$ time whether G admits an upward book embedding respecting $\mathcal{E}$. We assume that G is connected and refer to the full version for the case in which it is not.

We review a tool for testing whether G admits an upward planar drawing Γ respecting $\mathcal{E}$. By Theorem 1, the bimodality of $\mathcal{E}$ is necessary for the existence of Γ and can be tested in $O(n)$ time. Thus, the existence of Γ corresponds to the existence of an upward-consistent angle assignment λ for $\mathcal{E}$, which can be decided by the following strategy of Bertolazzi et al. [11]. Starting from the plane digraph G with planar embedding $\mathcal{E}$, one can construct a planar flow network $\mathcal{N}$, as illustrated in Fig. 6, where:

- for each switch v of G, the network $\mathcal{N}$ contains a source s_v supplying a single unit of flow;
- for each face f of $\mathcal{E}$, the network $\mathcal{N}$ contains a sink t_f that demands a number of units of flow equal to $n_f/2 - 1$ or $n_f/2 + 1$, depending on whether f is an internal face or the outer face of $\mathcal{E}$, respectively (where n_f is the number of switch angles incident to f);
- for each angle α in $\mathcal{E}$ at a switch of G, the network $\mathcal{N}$ contains a node w_α; and

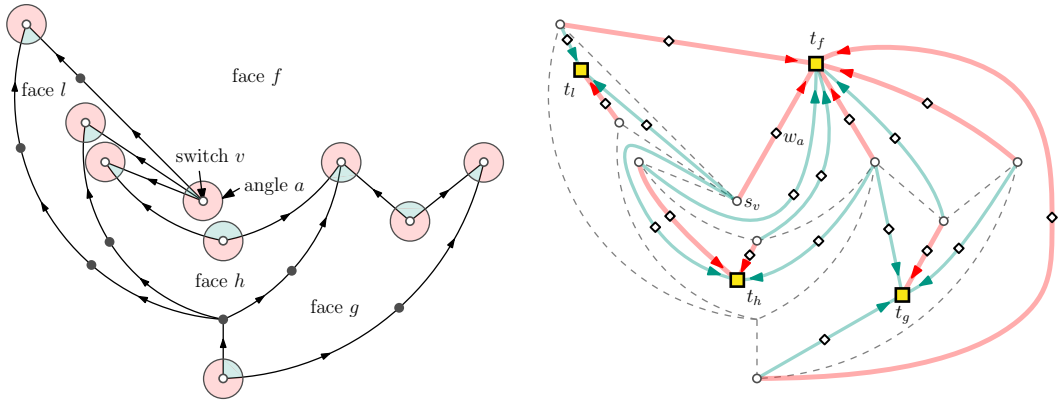


Figure 6 (a) An upward planar drawing of a digraph G; large and small angles at switches of G are depicted as red and green sectors of discs, respectively. (b) The network $\mathcal{N}$ constructed from the planar embedding of G; the edges of G, which are not part of $\mathcal{N}$, are drawn as gray dashed curves. Arcs of $\mathcal{N}$ traversed by the flow are red and thick, whereas arcs with no flow are green and thin.

- the network $\mathcal{N}$ contains an arc from each source s_v to each node w_α such that the angle α is incident to v in $\mathcal{E}$ and an arc from each node w_α to each sink t_f such that the angle α is incident to f in $\mathcal{E}$; all such arcs have a capacity of a single unit of flow.

By [11], there exists an upward-consistent angle assignment λ for $\mathcal{E}$ if and only if $\mathcal{N}$ admits a feasible flow whose value is the sum d_T of the demands of the sinks in $\mathcal{N}$. Indeed, upward-consistent angle assignments and (integral) feasible flows with value d_T are in bijection. Namely, from an upward-consistent angle assignment λ, one gets a feasible flow for $\mathcal{N}$ with value d_T by assigning flow 1 only to arcs incident to a node w_α where α is assigned a large angle by λ, and from a feasible flow for $\mathcal{N}$ with value d_T, one gets an upward-consistent angle assignment λ by assigning a large angle to each switch angle α such that the arcs incident to node w_α are assigned one unit of flow, and a small angle to all other switch angles. Testing whether G admits an upward planar drawing respecting $\mathcal{E}$ then becomes equivalent to testing whether $\mathcal{N}$ admits a flow whose value is d_T, which can be done in $O(n \log^3 n)$ time [17].

Our idea is to modify $\mathcal{N}$ so that there is an upward-consistent angle assignment λ for $\mathcal{E}$ such that $(\mathcal{E}, \lambda)$ is a good embedding if and only if $\mathcal{N}$ has a feasible flow whose value is d_T. The correspondence is actually stronger: As we show in the full version, the integral feasible flows with value d_T for the modified network are in bijection with the upward-consistent angle assignments λ for $\mathcal{E}$ such that $(\mathcal{E}, \lambda)$ is a good embedding. We show an algorithm, called $\mathcal{N}$-modifier, that performs a sequence of modifications to $\mathcal{N}$. Along the way, $\mathcal{N}$-modifier might stop and conclude that G has no upward book embedding respecting $\mathcal{E}$.

The $\mathcal{N}$-modifier algorithm. We start by describing the modifications to $\mathcal{N}$ to ensure 4-modality. Consider any vertex v of G. If v is not a switch, then its 4-modality does not depend on the angle assignment, thus we check whether in $\mathcal{E}$, in clockwise order around v, we have outgoing left edges, outgoing right edges, incoming right edges, and incoming left edges. If the test is negative, then G admits no upward book embedding respecting $\mathcal{E}$. If v is a switch, say a source, then we check whether in $\mathcal{E}$ the outgoing left edges are consecutive. If the test is negative, then G admits no upward book embedding respecting $\mathcal{E}$. If the test is positive and v has no outgoing left edges or no outgoing right edges, the processing of v is concluded. If the test is positive and v has both outgoing left and outgoing right edges, we modify $\mathcal{N}$ as follows; see Fig. 7. Let e be the left edge outgoing from v such that the edge e'

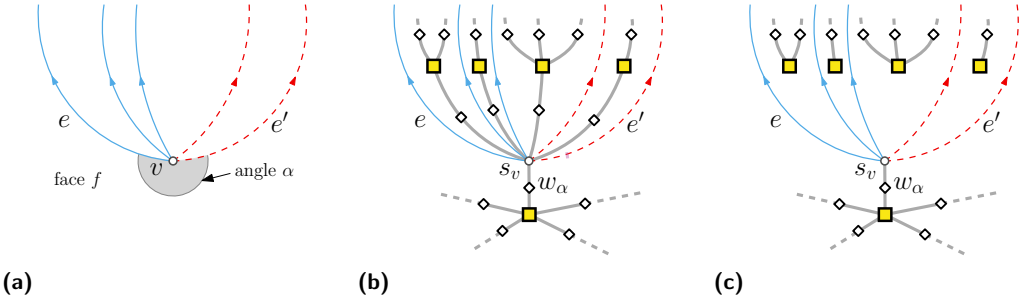


Figure 7 Modifying $\mathcal{N}$ to ensure 4-modality. (a) A vertex v with both outgoing left edges and outgoing right edges. (b) The part of $\mathcal{N}$ close to v (the edges of G are not in $\mathcal{N}$, but they are shown to maintain a visual reference with (a)). (c) Removal of the neighbors of s_v different from w_α.

preceding e in clockwise order around v is a right edge, let α be the angle (e, e'). For each angle $\beta \neq \alpha$ incident to v in $\mathcal{E}$, the algorithm $\mathcal{N}$-MODIFIER removes w_β and its incident arcs from $\mathcal{N}$. This forces the arc (s_v, w_α) to be assigned one unit of flow, hence making α large.

We next describe the modifications $\mathcal{N}$-MODIFIER applies to $\mathcal{N}$ in order to ensure the absence of impossible faces. Consider any face f of $\mathcal{E}$ and any maximal directed path ℓ_f in the boundary of f with f to its right (the treatment of the maximal directed paths with f to their left is analogous). If ℓ_f contains a left edge or if the rest of the boundary of f is a single edge, then the processing of ℓ_f is concluded. Otherwise, let u and v be the extremes of ℓ_f, let e_f^u and e_f^v be the edges of ℓ_f incident to u and v, let α_f^u be the angle in f incident to u and to the right of e_f^u, and let α_f^v be the angle in f incident to v and to the right of e_f^v. By Theorem 3, at least one of the angles α_f^u and α_f^v has to be large. By the maximality of ℓ_f, both α_f^u and α_f^v are switch angles. We say that α_f^u (resp. α_f^v) is ***enlargeable*** if the node $w_{\alpha_f^u}$ (resp. the node $w_{\alpha_f^v}$) is in $\mathcal{N}$. Note that, even if u (resp. v) is a switch of G, $w_{\alpha_f^u}$ (resp. $w_{\alpha_f^v}$) might not be in $\mathcal{N}$, because of some previous modification of $\mathcal{N}$. We distinguish three cases.

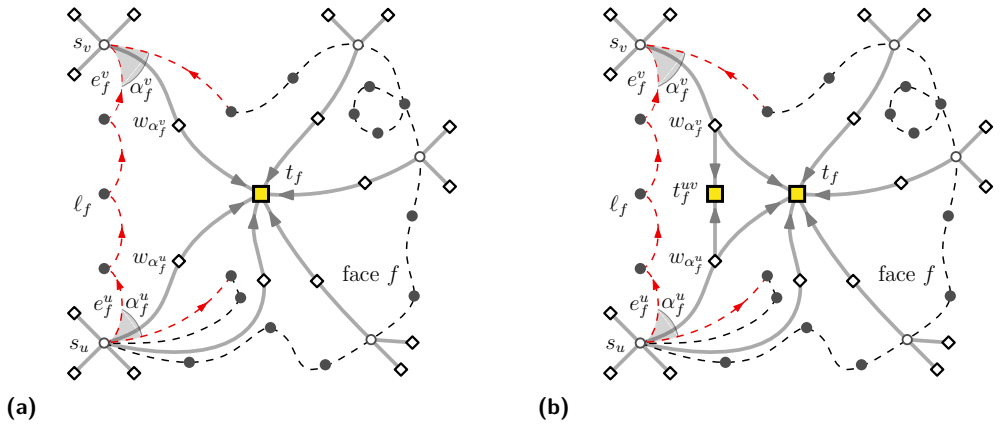


Figure 8 Modifying $\mathcal{N}$ to avoid impossible faces determined by ℓ_f, when both α_f^u and α_f^v are enlargeable. The part of $\mathcal{N}$ associated with f before (a) and after (b) the modification.

- If neither α_f^u nor α_f^v is enlargeable, then G admits no upward book embedding respecting $\mathcal{E}$.
- If just one of α_f^u and α_f^v, say α_f^u, is enlargeable, we modify $\mathcal{N}$ as follows: For each angle $\beta \neq \alpha_f^u$ incident to u in $\mathcal{E}$, we remove w_β and its incident arcs from $\mathcal{N}$, thus forcing the arc $(s_u, w_{\alpha_f^u})$ to be assigned one unit of flow and making α_f^u large.

- Finally, suppose that both α_f^u and α_f^v are enlargeable, as in Fig. 8. If there exists no angle $\beta \neq \alpha_f^u$ incident to u whose corresponding node w_β is in $\mathcal{N}$, or no angle $\beta \neq \alpha_f^v$ incident to v whose corresponding node w_β is in $\mathcal{N}$, then the processing of ℓ_f is concluded, since it is already guaranteed that one of α_f^u and α_f^v will be a large angle. Otherwise, we add to $\mathcal{N}$ a sink t_f^{uv} with demand 1, as well as arcs $(w_{\alpha_f^u}, t_f^{uv})$ and $(w_{\alpha_f^v}, t_f^{uv})$ with capacity 1, and decrease by 1 the demand of t_f. This forces one of $(w_{\alpha_f^u}, t_f^{uv})$ and $(w_{\alpha_f^v}, t_f^{uv})$ (and consequently one of $(s_u, w_{\alpha_f^u})$ and $(s_v, w_{\alpha_f^v})$) to be assigned one unit of flow, hence making large α_f^u or α_f^v, respectively. Note that $\mathcal{N}$ remains a planar flow network.

The described algorithm is the main ingredient for the proof of the following theorem.

▶ **Theorem 6.** *Let G be an n-vertex partitioned plane digraph with planar embedding $\mathcal{E}$. It is possible to test in $O(n \log^3 n)$ time whether G admits an upward book embedding respecting $\mathcal{E}$.*

Sketch. In the full version, we prove the mentioned bijection between the feasible flows with value d_T of the network $\mathcal{N}$ constructed by $\mathcal{N}$-MODIFIER and the upward-consistent angle assignments λ such that $(\mathcal{E}, \lambda)$ is a good embedding. Then Theorem 3 implies that G has an upward book embedding respecting $\mathcal{E}$ if and only if $\mathcal{N}$-MODIFIER did not conclude the opposite and $\mathcal{N}$ has a feasible flow with value d_T. Since $\mathcal{N}$-MODIFIER can be easily implemented to run in $O(n)$ time and the algorithm by Borradaile et al. [17] to test whether $\mathcal{N}$ has a feasible flow with value d_T runs in $O(n \log^3 n)$ time, the theorem follows. ◀

6 Test for Biconnected Partitioned Directed Partial 2-Trees

In this section we show how to test whether an n-vertex biconnected partitioned directed partial 2-tree $G = (V, L \cup R)$ admits an upward book embedding in two pages. We leverage tools introduced in [18, 19] to efficiently test whether a directed partial 2-tree is upward planar. Note that, lacking a characterization such as the one in Theorem 3, it would be prohibitive to lift such tools to work for our problem. Let e^* be an edge of G. We test in $O(n^2)$ time whether G admits a good embedding with e^* on the outer face. Repeating this test for all $O(n)$ choices of e^* yields an $O(n^3)$-time algorithm to decide whether G admits a good embedding. By Theorem 3, this is equivalent to testing whether G admits an upward book embedding.

Let T be an SPQ-tree of G, rooted at the Q-node ρ^* corresponding to e^*. Let μ be a node of T with poles u and v and with pertinent graph G_μ. A ***uv-external good embedding*** of G_μ is a good embedding of G_μ in which u and v are incident to the outer face. The requirement that e^* is incident to the outer face of any upward embedding $(\mathcal{E}, \lambda)$ of G implies that the restriction of $(\mathcal{E}, \lambda)$ to G_μ is indeed a uv-external good embedding of G_μ.

A uv-external good embedding $(\mathcal{E}_\mu, \lambda_\mu)$ of G_μ can be succinctly represented by a ***descriptor pair***, which consists of a ***shape descriptor*** $\langle \tau_l, \tau_r, \lambda^u, \lambda^v, \rho_l^u, \rho_r^u, \rho_l^v, \rho_r^v \rangle$ and of a ***pbe descriptor*** (short for partitioned book embedding descriptor) $\langle p_l^u, p_r^u, p_l^v, p_r^v, \chi_l, \chi_r, \alpha_l^u, \alpha_r^u, \alpha_l^v, \alpha_r^v \rangle$, with the following meaning. Let the ***left outer path*** P_l (resp. the ***right outer path*** P_r) be the path obtained by traversing the boundary of the outer face f_μ of $\mathcal{E}_\mu$ from u to v in clockwise (resp. counterclockwise) direction. The label τ_l (τ_r) is the sum of the labels assigned by λ_μ to the angles in f_μ at the vertices of P_l (resp. P_r), excluding u and v. The values λ^u and λ^v are the labels of the angles at u and v in f_μ, respectively. The label ρ_l^u is `in` or `out` depending on whether the edge of P_l incident to u is incoming or outgoing at u, respectively; the values ρ_r^u, ρ_l^v, and ρ_r^v are defined analogously. The label p_l^u is L or R depending on whether the edge of P_l incident to u belongs to L or R, respectively; the values p_r^u, p_l^v, and p_r^v are defined analogously. The label χ_l is 1 if P_l is a directed path from u to v and all its edges belong to L or if P_l is a directed path from v to u and all its

edges belong to R, it is 0 otherwise. The label χ_r is defined analogously for P_r. The label α_l^u is 1 if P_l contains a directed path from u to a vertex $w \notin \{u, v\}$ whose edges belong to L or a directed path from a vertex $w \notin \{u, v\}$ to u whose edges belong to R and if λ_μ assigns a small angle at w in f_μ; the label α_l^u is 0 otherwise. The labels α_r^u, α_l^v, and α_r^v are defined analogously, with respect to P_r rather than P_l and/or with respect to v rather than u.

The information in a descriptor pair fully describes how a uv-external good embedding of G_μ interfaces with the rest of G for the construction of a good embedding of G with e^* on the outer face. That is, for a good embedding $(\mathcal{E}, \lambda)$ of G with e^* on the outer face, let $(\mathcal{E}_\mu, \lambda_\mu)$ be the uv-external good embedding of G_μ in $(\mathcal{E}, \lambda)$, and let (σ, ω) be the descriptor pair of $(\mathcal{E}_\mu, \lambda_\mu)$. Replacing $(\mathcal{E}_\mu, \lambda_\mu)$ with *any* other uv-external good embedding of G_μ with descriptor pair (σ, ω) still results in a good embedding of G with e^* on the outer face. Even more, for a uv-external good embedding $(\mathcal{E}_\mu, \lambda_\mu)$ of G_μ with descriptor pair (σ, ω), let ν be a child of μ with poles w and z, and let (σ', ω') be the descriptor pair of the wz-external good embedding $(\mathcal{E}_\nu, \lambda_\nu)$ of G_ν in $(\mathcal{E}_\mu, \lambda_\mu)$. Replacing $(\mathcal{E}_\nu, \lambda_\nu)$ with *any* other wz-external good embedding of G_ν with descriptor pair (σ', ω') results in a uv-external good embedding of G_μ *with descriptor pair* (σ, ω). This allows us to only keep track of the descriptor pairs (σ, ω) that are "realizable" by G_μ, rather than of the actual uv-external good embeddings of G_μ. This is formalized by the notion of ***feasible set*** $\mathcal{F}_\mu$ for μ, which contains all the descriptor pairs (σ, ω) such that G_μ admits a uv-external good embedding with descriptor pair (σ, ω).

The core of our algorithm consists of a bottom-up computation of the feasible set $\mathcal{F}_\mu$ for each node μ of T. If μ is a **non-root Q-node**, then G_μ has a unique uv-external good embedding, hence $\mathcal{F}_\mu$ contains a unique descriptor pair and is computed in $O(1)$ time.

If μ is an **S-node**, let ν_1 and ν_2 be the children of μ in T and let w be the unique vertex shared by G_{ν_1} and G_{ν_2}. We combine every descriptor pair (σ_1, ω_1) in $\mathcal{F}_{\nu_1}$ with every descriptor pair (σ_2, ω_2) in $\mathcal{F}_{\nu_2}$; for every such combination, we assign the two angles at w in the outer face with every possible label in $\{-1, 0, 1\}$. We test in $O(1)$ time whenever the combination and the assignment result in a descriptor pair (σ, ω) of a good embedding of G_μ, by checking whether the properties of Theorems 1 and 3 are satisfied. In the positive case, we add (σ, ω) to $\mathcal{F}_\mu$. Thus, the computation of $\mathcal{F}_\mu$ takes time $O(|\mathcal{F}_{\nu_1}| \cdot |\mathcal{F}_{\nu_2}|)$, which is in $O(|V(G_{\nu_1})| \cdot |V(G_{\nu_2})|)$. This sums up to $O(n^2)$ time over all S-nodes of T.

If μ is a **P-node** with children $\nu_1, \ldots, \nu_k$, we cannot just combine the descriptor pairs in $\mathcal{F}(\nu_1), \ldots, \mathcal{F}(\nu_k)$, as the number of combinations might be super-polynomial. Instead, we consider every descriptor pair that *might* describe a uv-external good embedding of G_μ and test whether it is in $\mathcal{F}_\mu$. This is done as follows. A set $\mathcal{U}_n$ of descriptor pairs is ***n-universal*** if: (i) for every h-vertex biconnected partitioned directed partial 2-tree H with $h \leq n$, for every two vertices u_H and v_H of H, and for every $u_H v_H$-external good embedding $(\mathcal{E}_H, \lambda_H)$ of H, the descriptor pair of $(\mathcal{E}_H, \lambda_H)$ is in $\mathcal{U}_n$, and (ii) for every descriptor pair (σ, ω) in $\mathcal{U}_n$, there exists a biconnected partitioned directed partial 2-tree H that contains two vertices u_H and v_H and admits a $u_H v_H$-external good embedding $(\mathcal{E}_H, \lambda_H)$ with descriptor pair (σ, ω). Note that $\mathcal{F}_\mu \subseteq \mathcal{U}_n$. We can construct an n-universal set $\mathcal{U}_n$ with $|\mathcal{U}_n| \in O(n)$ in $O(n)$ time.

Consider a uv-external good embedding $(\mathcal{E}_\mu, \lambda_\mu)$ of G_μ with descriptor pair (σ, ω). Assume, w.l.o.g. up to a change of the indices, that the clockwise order around u in $\mathcal{E}_\mu$ of the pertinent graphs of the children of μ starting at the outer face is $G_{\nu_1}, \ldots, G_{\nu_k}$. The ***descriptor sequence*** of $(\mathcal{E}_\mu, \lambda_\mu)$ is the sequence $\mathcal{S}_\mu = [(\sigma_1, \omega_1), \ldots, (\sigma_k, \omega_k)]$ of the descriptor pairs of the uv-external good embeddings of $G_{\nu_1}, \ldots, G_{\nu_k}$ in $(\mathcal{E}_\mu, \lambda_\mu)$. The ***contracted descriptor sequence*** of $(\mathcal{E}_\mu, \lambda_\mu)$ is obtained from $\mathcal{S}_\mu$ by identifying equal consecutive descriptor pairs, see Fig. 9. Finally, the ***generating set*** $\mathcal{G}(\sigma, \omega)$ for (σ, ω) is the set of contracted descriptor

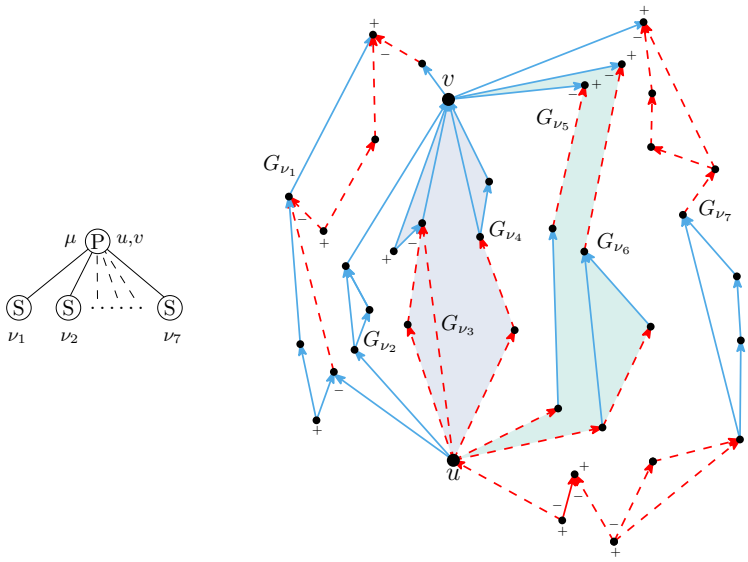


Figure 9 A uv-external good embedding $(\mathcal{E}_\mu, \lambda_\mu)$ of G_μ. Large and small angles on the outer faces of the embeddings of $G_{\nu_1}, G_{\nu_2}, \ldots, G_{\nu_7}$ are marked $+$ and $-$, respectively. The descriptor pair (σ, ω) of $(\mathcal{E}_\mu, \lambda_\mu)$ has $\sigma = \langle 1, 2, 0, -1, \texttt{out}, \texttt{in}, \texttt{out}, \texttt{out}\rangle$ and $\omega = \langle L, R, L, L, 0, 0, 1, 0, 0, 0\rangle$. The descriptor sequence of $(\mathcal{E}_\mu, \lambda_\mu)$ is $[(\sigma_1, \omega_1), (\sigma_2, \omega_2), \ldots, (\sigma_7, \omega_7)]$, where $\sigma_1 = \langle 1, -1, 1, 1, \texttt{out}, \texttt{out}, \texttt{out}, \texttt{out}\rangle$, $\sigma_2 = \langle 0, 0, 1, 1, \texttt{out}, \texttt{out}, \texttt{in}, \texttt{in}\rangle$, $\sigma_3 = \sigma_4 = \langle 0, 0, 1, 1, \texttt{out}, \texttt{out}, \texttt{in}, \texttt{in}\rangle$, $\sigma_5 = \sigma_6 = \langle -1, 1, 1, 1, \texttt{out}, \texttt{out}, \texttt{out}, \texttt{out}\rangle$, $\sigma_7 = \langle -2, 2, 1, 1, \texttt{in}, \texttt{in}, \texttt{out}, \texttt{out}\rangle$, $\omega_1 = \langle L, L, L, L, 0, 0, 1, 0, 0, 0\rangle$, $\omega_2 = \langle L, L, L, L, 1, 0, 0, 0, 0, 0\rangle$, $\omega_3 = \omega_4 = \omega_5 = \omega_6 = \langle R, R, L, L, 0, 0, 0, 0, 0, 0\rangle$, and $\omega_7 = \langle R, R, L, L, 0, 0, 1, 0, 0, 0\rangle$. The contracted descriptor sequence of $(\mathcal{E}_\mu, \lambda_\mu)$ is $[(\sigma_1, \omega_1), (\sigma_2, \omega_2), (\sigma_3, \omega_3), (\sigma_5, \omega_5), (\sigma_7, \omega_7)]$.

sequences that a P-node with poles w and z can have in a wz-external good embedding with descriptor pair (σ, ω). We can prove that $\mathcal{G}(\sigma, \omega)$ has size $O(1)$, can be constructed in $O(1)$ time, and that each contracted descriptor sequence in $\mathcal{G}(\sigma, \omega)$ has length $O(1)$.

In order to compute $\mathcal{F}_\mu$, we construct the generating set $\mathcal{G}(\sigma, \omega)$ of (σ, ω), for each descriptor pair (σ, ω) in $\mathcal{U}_n$, in overall $O(n)$ time. For each contracted descriptor sequence $\mathcal{C}$ in $\mathcal{G}(\sigma, \omega)$, we test whether $\mathcal{C}$ is ***realizable*** by G_μ, i.e., whether there exists a uv-external good embedding of G_μ whose contracted descriptor sequence $\mathcal{C}'$ is a subsequence of $\mathcal{C}$ containing the first and last elements of $\mathcal{C}$, which ensures that $\mathcal{C}'$ belongs to $\mathcal{G}(\sigma, \omega)$. We decide whether $\mathcal{C}$ is realizable by G_μ in $O(k)$ time, by constructing an auxiliary graph with vertex set $(\{\nu_1, \ldots, \nu_k\}, \{(\sigma_1, \omega_1), \ldots, (\sigma_\ell, \omega_\ell)\})$ and with an edge between ν_i and (σ_j, ω_j) if (σ_j, ω_j) belongs to $\mathcal{F}_{\nu_i}$, and by then testing whether each vertex in $\{\nu_1, \ldots, \nu_k\}$ can be assigned to a neighbor, so that both (σ_1, ω_1) and $(\sigma_\ell, \omega_\ell)$ are assigned at least one vertex. We add (σ, ω) to $\mathcal{F}_\mu$ if some $\mathcal{C}$ in $\mathcal{G}(\sigma, \omega)$ is realizable by G_μ. Since $|\mathcal{U}_n| \in O(n)$, $|\mathcal{G}(\sigma, \omega)| \in O(1)$, and the realizability of a contracted descriptor sequence can be decided in $O(k)$ time, the computation of $\mathcal{F}_\mu$ takes $O(nk)$ time, which sums up to $O(n^2)$ time over all P-nodes of T.

Finally, if μ is the **root Q-node** ρ^*, we treat it as a P-node with two children, corresponding to e^* and $G - e^*$, and compute $\mathcal{F}_{\rho^*}$ in $O(n)$ time. Then G has a good embedding with e^* on the outer face if and only if $\mathcal{F}_{\rho^*}$ is non-empty. We thus get the following.

▶ **Theorem 7.** *Let G be an n-vertex biconnected partitioned directed partial 2-tree. It is possible to test in $O(n^3)$ time whether G admits an upward book embedding.*

7 Conclusions

We studied upward book embeddings of partitioned digraphs, focusing on the unsolved case of two pages – one of the final open algorithmic book-embedding problems. We closed the complexity gap for the problem by proving its NP-hardness. We provided a characterization of the upward embeddings that admit such layouts and combined this with several algorithmic tools, including flow techniques, SPQ-decompositions, and compact embedding representations, obtaining efficient testing and embedding algorithms for digraphs with a fixed planar embedding and for biconnected partial 2-trees with variable embedding. We aim to (i) remove the linear overhead in our algorithm for biconnected partial 2-trees caused by rerouting the SPQ-tree at each Q-node, and (ii) extend the method to arbitrary simply connected partial 2-trees. The latter is challenging due to nested biconnected components in the block-cut-vertex tree, which must be arranged while keeping cut vertices 4-modal and avoiding impossible faces. Additional research directions include:

- studying the complexity of the problem for partitioned digraphs with few sources;
- determining whether the problem is NP-complete for bounded-treewidth instances; and
- developing FPT algorithms for parameters more restrictive than treewidth.

References

1 Hugo A. Akitaya, Erik D. Demaine, Adam Hesterberg, and Quanquan C. Liu. Upward partitioned book embeddings. In Fabrizio Frati and Kwan-Liu Ma, editors, *25th International Symposium on Graph Drawing and Network Visualization (GD'17)*, volume 10692 of *LNCS*, pages 210–223. Springer, 2017. doi:10.1007/978-3-319-73915-1_18.

2 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, and Fabrizio Frati. Strip planarity testing for embedded planar graphs. *Algorithmica*, 77(4):1022–1059, 2017. doi:10.1007/S00453-016-0128-9.

3 Patrizio Angelini, Giordano Da Lozzo, and Daniel Neuwirth. Advancements on SEFE and partitioned book embedding problems. *Theor. Comput. Sci.*, 575:71–89, 2015. doi:10.1016/j.tcs.2014.11.016.

4 Patrizio Angelini, Marco Di Bartolomeo, and Giuseppe Di Battista. Implementing a partitioned 2-page book embedding testing algorithm. In Walter Didimo and Maurizio Patrignani, editors, *20th International Symposium on Graph Drawing (GD'12)*, volume 7704 of *LNCS*, pages 79–89. Springer, 2012. doi:10.1007/978-3-642-36763-2_8.

5 Gail H. Atneosen. One-dimensional n-leaved continua. *Fund. Math.*, 74(1):43–45, 1972. doi:10.4064/fm-74-1-43-45.

6 Patricia Bachmann, Ignaz Rutter, and Peter Stumpf. On 3-coloring circle graphs. In Michael A. Bekos and Markus Chimani, editors, *31st International Symposium on Graph Drawing and Network Visualization, (GD'23)*, volume 14465 of *LNCS*, pages 152–160. Springer, 2023. doi:10.1007/978-3-031-49272-3_11.

7 Patricia Bachmann, Ignaz Rutter, and Peter Stumpf. On 3-coloring circle graphs. *J. Graph Algorithms Appl.*, 28(1):389–402, 2024. doi:10.7155/jgaa.v28i1.2991.

8 Michael A. Bekos, Giordano Da Lozzo, Fabrizio Frati, Martin Gronemann, Tamara Mchedlidze, and Chrysanthi N. Raftopoulou. Recognizing DAGs with page-number 2 is NP-complete. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium on Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *LNCS*, pages 361–370. Springer, 2022. doi:10.1007/978-3-031-22203-0_26.

9 Michael A. Bekos, Giordano Da Lozzo, Fabrizio Frati, Martin Gronemann, Tamara Mchedlidze, and Chrysanthi N. Raftopoulou. Recognizing DAGs with page-number 2 is NP-complete. *Theor. Comput. Sci.*, 946:113689, 2023. doi:10.1016/J.TCS.2023.113689.

10 Frank Bernhart and Paul C Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.

11 Paola Bertolazzi, Giuseppe Di Battista, Giuseppe Liotta, and Carlo Mannino. Upward drawings of triconnected digraphs. *Algorithmica*, 12(6):476–497, 1994. doi:10.1007/BF01188716.

12 Paola Bertolazzi, Giuseppe Di Battista, Carlo Mannino, and Roberto Tamassia. Optimal upward planarity testing of single-source digraphs. *SIAM J. Comput.*, 27(1):132–169, 1998. doi:10.1137/S0097539794279626.

13 Sandeep N. Bhatt, Fan R.K. Chung, F.Thomas Leighton, and Arnold L. Rosenberg. Scheduling tree-dags using FIFO queues: A control–memory trade-off. *Journal of Parallel and Distributed Computing*, 33(1):55–68, 1996. doi:10.1006/jpdc.1996.0024.

14 T. Biedl, T. Shermer, S. Whitesides, and S. Wismath. Bounds for orthogonal 3-D graph drawing. *Journal of Graph Algorithms and Applications*, 3(4):63–79, January 1999. doi:10.7155/jgaa.00018.

15 Carla Binucci, Giordano Da Lozzo, Emilio Di Giacomo, Walter Didimo, Tamara Mchedlidze, and Maurizio Patrignani. Upward book embeddings of st-graphs. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry (SOCG'19)*, volume 129 of *LIPIcs*, pages 13:1–13:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.SoCG.2019.13.

16 Carla Binucci, Giordano Da Lozzo, Emilio Di Giacomo, Walter Didimo, Tamara Mchedlidze, and Maurizio Patrignani. Upward book embeddability of st-graphs: Complexity and algorithms. *Algorithmica*, 85(12):3521–3571, 2023. doi:10.1007/s00453-023-01142-y.

17 Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM J. Comput.*, 46(4):1280–1303, 2017. doi:10.1137/15M1042929.

18 Steven Chaplick, Emilio Di Giacomo, Fabrizio Frati, Robert Ganian, Chrysanthi N. Raftopoulou, and Kirill Simonov. Parameterized algorithms for upward planarity. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry (SOCG'22)*, volume 224 of *LIPIcs*, pages 26:1–26:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SOCG.2022.26.

19 Steven Chaplick, Emilio Di Giacomo, Fabrizio Frati, Robert Ganian, Chrysanthi N. Raftopoulou, and Kirill Simonov. Testing upward planarity of partial 2-trees. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium on Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *LNCS*, pages 175–187. Springer, 2022. doi:10.1007/978-3-031-22203-0_13.

20 Fan R. K. Chung, Frank Thomson Leighton, and Arnold L. Rosenberg. Embedding graphs in books: A layout problem with applications to VLSI design. *SIAM Journal on Algebraic Discrete Methods*, 8(1):33–58, 1987. doi:10.1137/0608002.

21 Giordano Da Lozzo, Fabrizio Frati, and Ignaz Rutter. Upward book embeddings of partitioned digraphs, 2026. URL: https://arxiv.org/abs/2603.17128.

22 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

23 Giuseppe Di Battista and Roberto Tamassia. Algorithms for plane representations of acyclic digraphs. *Theor. Comput. Sci.*, 61:175–198, 1988. doi:10.1016/0304-3975(88)90123-5.

24 Giuseppe Di Battista and Roberto Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25(5):956–997, 1996. doi:10.1137/S0097539794280736.

25 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, and Stephen K. Wismath. Book embeddability of series-parallel digraphs. *Algorithmica*, 45(4):531–547, 2006. doi:10.1007/s00453-005-1185-7.

26 Walter Didimo, Francesco Giordano, and Giuseppe Liotta. Upward spirality and upward planarity testing. *SIAM J. Discret. Math.*, 23(4):1842–1899, 2009. doi:10.1137/070696854.

27 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

28 Vida Dujmović and David R. Wood. On linear layouts of graphs. *Discret. Math. Theor. Comput. Sci.*, 6(2):339–358, 2004. doi:10.46298/dmtcs.317.

29 Hazel Everett, Sylvain Lazard, Giuseppe Liotta, and Stephen Wismath. Universal sets of n points for one-bend drawings of planar graphs with n vertices. *Discret. Comput. Geom.*, 43(2):272–288, 2010. doi:10.1007/s00454-009-9149-3.

30 Fabrizio Frati, Joachim Gudmundsson, and Emo Welzl. On the number of upward planar orientations of maximal planar graphs. *Theor. Comput. Sci.*, 544:32–59, 2014. doi:10.1016/J.TCS.2014.04.024.

31 Robert Ganian, Haiko Müller, Sebastian Ordyniak, Giacomo Paesani, and Mateusz Rychlicki. A tight subexponential-time algorithm for two-page book embedding. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming (ICALP'24)*, volume 297 of *LIPIcs*, pages 68:1–68:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICALP.2024.68.

32 Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.*, 31(2):601–625, 2001. doi:10.1137/S0097539794277123.

33 F. Giordano, G. Liotta, T. Mchedlidze, A. Symvonis, and S.H. Whitesides. Computing upward topological book embeddings of upward planar digraphs. *Journal of Discrete Algorithms*, 30:45–69, 2015. doi:10.1016/j.jda.2014.11.006.

34 Carsten Gutwenger and Petra Mutzel. A linear time implementation of SPQR-trees. In Joe Marks, editor, *8th International Symposium on Graph Drawing (GD'00)*, volume 1984 of *LNCS*, pages 77–90. Springer, 2000. doi:10.1007/3-540-44541-2_8.

35 Lenwood S. Heath and Sriram V. Pemmaraju. Stack and queue layouts of directed acyclic graphs: Part II. *SIAM J. Comput.*, 28(5):1588–1626, 1999. doi:10.1137/S0097539795291550.

36 Lenwood S Heath, Sriram V Pemmaraju, and Calvin J Ribbens. *Sparse matrix-vector multiplication on a small linear array.* University of Iowa, Department of Computer Science, 1993.

37 Lenwood S. Heath, Sriram V. Pemmaraju, and Ann N. Trenk. Stack and queue layouts of directed acyclic graphs: Part I. *SIAM J. Comput.*, 28(4):1510–1539, 1999. doi:10.1137/S0097539795280287.

38 Seok-Hee Hong and Hiroshi Nagamochi. Two-page book embedding and clustered graph planarity. Technical report, Dept. of Applied Mathematics and Physics, University of Kyoto, Japan, 2009.

39 Seok-Hee Hong and Hiroshi Nagamochi. Simpler algorithms for testing two-page book embedding of partitioned graphs. *Theor. Comput. Sci.*, 725:79–98, 2018. doi:10.1016/J.TCS.2015.12.039.

40 G. Jacobson. Space-efficient static trees and graphs. In *30th Annual Symposium on Foundations of Computer Science (FOCS '89)*, pages 549–554. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63533.

41 Bart M. P. Jansen, Liana Khazaliya, Philipp Kindermann, Giuseppe Liotta, Fabrizio Montecchiani, and Kirill Simonov. Upward and orthogonal planarity are W[1]-hard parameterized by treewidth. In Michael A. Bekos and Markus Chimani, editors, *31st International Symposium on Graph Drawing and Network Visualization (GD'23)*, volume 14466 of *LNCS*, pages 203–217. Springer, 2023. doi:10.1007/978-3-031-49275-4_14.

42 Tamara Mchedlidze and Antonios Symvonis. Crossing-free acyclic Hamiltonian path completion for planar st-digraphs. In Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors, *20th International Symposium on Algorithms and Computation (ISAAC'09)*, volume 5878 of *LNCS*, pages 882–891. Springer, 2009. doi:10.1007/978-3-642-10631-6_89.

43 Tamara Mchedlidze and Antonios Symvonis. Crossing-optimal acyclic HP-completion for outerplanar st-digraphs. *J. Graph Algorithms Appl.*, 15(3):373–415, 2011. doi:10.7155/jgaa.00231.

44 Kurt Mehlhorn. *Data Structures and Algorithms: Multi-dimensional Searching and Computational Geometry*, volume 3 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1984. doi:10.1007/978-3-642-69900-9.

45 T. D. Morgan. *Map folding*. PhD thesis, Massachusetts Institute of Technology, 2012. Ph.D. thesis. URL: https://hdl.handle.net/1721.1/77030.

46 J. Ian Munro and Venkatesh Raman. Succinct representation of balanced parentheses and static trees. *SIAM J. Comput.*, 31(3):762–776, 2001. doi:10.1137/S0097539799364092.

47 T. Ollmann. On the book thicknesses of various graphs. In F. Hoffman, R.B. Levow, and R.S.D. Thomas, editors, *Southeastern Conference on Combinatorics, Graph Theory and Computing*, volume VIII of *Congressus Numerantium*, page 459, 1973.

48 C. A. Persinger. Subsets of n-books in E^3. *Pacific J. Math.*, 18:169–173, 1966. URL: http://projecteuclid.org/euclid.pjm/1102994417.

49 Vaughan R. Pratt. Computing permutations with double-ended queues, parallel stacks and parallel queues. In *5th Symposium on Theory of Computing (STOC'73)*, pages 268–277. ACM, 1973. doi:10.1145/800125.804058.

50 Arnold L. Rosenberg. The diogenes approach to testable fault-tolerant arrays of processors. *IEEE Trans. Computers*, 32(10):902–910, 1983. doi:10.1109/TC.1983.1676134.

51 Ajaykrishnan E S, Robert Ganian, Daniel Lokshtanov, and Vaishali Surianarayanan. A quasi-polynomial time algorithm for 3-coloring circle graphs. In Sepehr Assadi and Eva Rotenberg, editors, *9th Symposium on Simplicity in Algorithms (SOSA'26)*, pages 65–80. SIAM, 2026. doi:10.1137/1.9781611978964.6.

52 Robert Tarjan. Sorting using networks of queues and stacks. *J. ACM*, 19(2):341–346, April 1972. doi:10.1145/321694.321704.

53 Walter Unger. On the k-colouring of circle-graphs. In Robert Cori and Martin Wirsing, editors, *5th Annual Symposium on Theoretical Aspects of Computer Science (STACS '88)*, volume 294 of *LNCS*, pages 61–72. Springer, 1988. doi:10.1007/BFb0035832.

54 Manfred Wiegers. Recognizing outerplanar graphs in linear time. In Gottfried Tinhofer and Gunther Schmidt, editors, *13th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'87)*, pages 165–176. Springer, 1987. doi:10.1007/3-540-17218-1_57.

55 Avi Wigderson. The complexity of the Hamiltonian circuit problem for maximal planar graphs. Technical Report TR-298, EECS Department, Princeton University, 1982. URL: https://www.math.ias.edu/avi/node/820.

A Free Lunch: Manifolds of Positive Reach Can Be Smoothed Without Decreasing the Reach

Hana Dal Poz Kouřimská
University of Potsdam, Germany

André Lieutier
Aix-en-Provence, France

Mathijs Wintraecken
Inria Centre d'Université Côte d'Azur, Sophia Antipolis, France

Abstract

Assumptions on the reach are crucial for ensuring the correctness of many geometric and topological algorithms, including triangulation, manifold reconstruction and learning, homotopy reconstruction, and methods for estimating curvature or reach. However, these assumptions are often coupled with the requirement that the manifold be smooth, typically at least C^2.

In this paper, we prove that any manifold with positive reach can be approximated arbitrarily well by a C^∞ manifold without significantly reducing the reach. More precisely, given a manifold with reach R, we construct a manifold that is ε-close to it in the C^1 sense (both the manifold and its tangent spaces are close), and has reach at least $R - \varepsilon$. The proof employs techniques from differential topology – partitions of unity and smoothing using convolution kernels.

This result implies that nearly all theorems established for C^2 or manifolds with a certain reach naturally extend to manifolds with the same reach, even if they are not C^2, for free!

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Reach, Manifolds, Smoothing, Differentiability, Differential topology

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.37

Related Version *Full Version*: `https://inria.hal.science/hal-04816535`

Funding *Hana Dal Poz Kouřimská*: Supported by the DFG project No. 524578210.
Mathijs Wintraecken: Supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 754411, the Austrian science fund (FWF) grant No. M-3073, the ANR grant StratMesh (ANR-24-CE48-1899), and the welcome package from IDEX of the Université Côte d'Azur, (ANR-15-IDEX-01).

Acknowledgements We thank Jean-Daniel Boissonnat for discussion. We would also like to acknowledge the organizers of the workshop on "Algorithms for the Medial Axis", and Erin Chambers in particular, for giving an impulse to this research. We further thank Victor Bangert for encouragement.

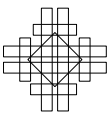
1 Introduction

What is the reach? The *reach* of a set is a number that captures the geometric properties of its *shape*. Roughly speaking, it provides a bound on the set's curvature and quantifies how far apart different parts of the set are from each other. As a key descriptor of a shape's complexity, the reach plays a crucial role as an assumption in many geometric and topological algorithms.

Formally, the reach of a (closed) set $\mathcal{S} \subset \mathbb{R}^d$ is the minimum of the distance between $\mathcal{S}$ and its *medial axis*, that is, the set of points in $\mathbb{R}^d$ for which the closest point in $\mathcal{S}$ is not unique. We illustrate these notions in Figure 1.



© Hana Dal Poz Kouřimská, André Lieutier, and Mathijs Wintraecken;
licensed under Creative Commons License CC-BY 4.0



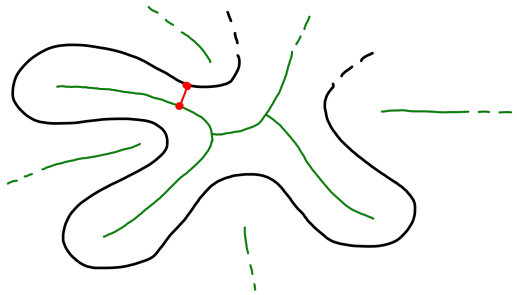


Figure 1 The medial axis (green) of a curve (black) in the plane. The reach is indicated in red.

The early history of the reach. The reach was first introduced by Federer in [20]. Notably, earlier work by Erdős explored what we now refer to as the *medial axis*, although it did not address the reach itself [18, 19]. While Erdős studied the medial axis and Federer considered its complement, the term "medial axis" itself was coined only later, by Blum [12]. A related notion, the *cut locus* in Riemannian geometry, has a significantly longer history, with its origins traced to the work of Poincaré [34], Whitehead [38], and Myers [31, 32][1].

Due to their wide applicability, these concepts have been reintroduced multiple times. For instance, the (closure of) medial axis was reintroduced as the *central set* by Milman and Waksman [30], its complement as the *unique footprint set* by Kleinjohann [25], and the reach was referred to as the *condition number* by Niyogi, Smale, and Weinberger [33].

The reach and differentiability. In [20], Federer established that the reach is stable under $C^{1,1}$-diffeomorphisms of the ambient space. Here, $C^{1,1}$ denotes a C^1 map whose derivative is Lipschitz, and by a $C^{1,1}$-diffeomorphism, we mean that both the diffeomorphism and its inverse are $C^{1,1}$. Federer also mentioned, without extensive detail [20, Remark 4.20], that the graph of a function has positive reach if and only if the function itself is $C^{1,1}$. Lytchak [27, 28] later proved that a topological submanifold of the Euclidean space without boundary has positive reach if and only if it is a $C^{1,1}$-submanifold. A quantified version of this statement can be found in [26].

The reach in computational geometry and topology, and manifold learning. The reach encapsulates the geometric complexity of a shape in a single non-negative value, making it a crucial assumption for ensuring the correctness of many geometric and topological algorithms. Several key classes of algorithms that depend on reach assumptions include:

- *Triangulation algorithms for surfaces and manifolds* (see for example [7, 6, 8, 13, 14, 15]).
- *Manifold learning or reconstruction and manipulation* (see for example [3, 4, 10, 22, 23, 36, 29, 5]).
- *Homotopy inference* (see for example [33, 37]).

All these algorithms have as input a point cloud queried from a manifold. Their output, and its correctness, varies:

- *Triangulation algorithms* produce as output a simplicial complex – often, but not always a variant of the Delaunay triangulation. Such algorithms are perceived as correct if the output complex is (piecewise linearly) homeomorphic to the underlying manifold.

[1] See [35] for a nice overview of the early history of the cut locus.

- *Manifold learning algorithms* output a description of an approximation of the given manifold – for example as the zero set of a function or a projection operator. Such algorithms are correct if the output approximation is homeomorphic to the underlying manifold and sufficiently geometrically close to it.
- *Homotopy inference algorithms* output a simplicial complex, which is correct if it is homotopy equivalent to the underlying manifold.

For the success of these algorithms, it is essential to provide criteria that guarantee their correctness and quality. Such criteria are often formulated using the reach of the manifold.

In addition, *reach estimation* in and by itself is an important topic in inference (see for example [1, 2, 11]). The typical setting for this type of problems involves a point cloud, which is drawn from a probability measure whose support is a manifold satisfying a regularity condition. Such condition can, for example, be a guarantee that the reach of the manifold in question is larger than some (very small positive) constant. The goal is to estimate the actual reach of the manifold, with a guaranteed precision margin.

Most of the aforementioned works assume that **the manifold in question is at least C^2**, in addition to having positive reach. This assumption is often made because it allows the use of the full machinery of differential and Riemannian geometry. For instance, the second fundamental form is always well-defined in the C^2 setting [16, 17]. Yet, this condition is not entirely natural. Manifolds with positive reach are indeed at least $C^{1,1}$, see [26, 27, 28], and by Rademacher's theorem [21], they are C^2 almost everywhere, but not necessarily everywhere.

On the other hand, requiring the manifold in question to be $C^{1,1}$ is quite natural, as it encompasses many configurations commonly found in modern modeling software, such as computer-aided design (CAD). This is because the majority of manufactured objects can be modeled as $C^{1,1}$ surfaces: For instance, consider a line segment and a circular arc intersecting at a point where their tangents coincide, as illustrated in Figure 2. This configuration has positive reach but is only $C^{1,1}$.



Figure 2 A $C^{1,1}$ transition (in red) between a circular arc and a straight line segment.

As a consequence of the results presented in this article (primarily Theorem 1), all aforementioned results can be extended from C^2 manifolds with positive reach to arbitrary manifolds with positive reach; hence the paper title.

Our settings. In this work, we identify an embedded manifold locally (in a neighborhood of a point p) as a graph of a map from its tangent space to its normal space at p. This approach, which we illustrate in Figure 3, allows us to compare homeomorphic manifolds ($\mathcal{M}$ and $\mathcal{M}'$ in Figure 3), embedded in Euclidean space of the same dimension, by comparing their corresponding maps and their derivatives (f and F in the figure). We say that two such manifolds are close *in the C^1 sense*, if, roughly speaking, both the manifolds and their tangent spaces are close. Formally, this is a condition on the corresponding maps f and F, and we explain it in Definition 11.

Throughout the paper, we denote the dimension of the manifolds $\mathcal{M}$ and $\mathcal{M}'$ by n, and the dimension of the ambient Euclidean space by d.

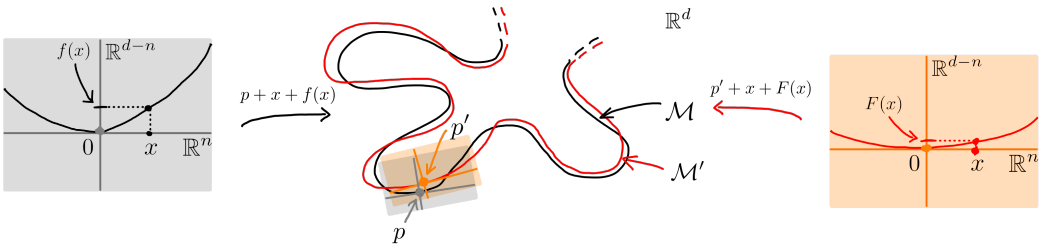


Figure 3 In our setting, we view manifolds locally as graphs of functions.

Our contribution. Our main contribution is the following statement:

▶ **Theorem 1.** *Let $\mathcal{M} \subset \mathbb{R}^d$ be a compact manifold of (positive) reach R, and $\varepsilon > 0$. Then there exists a C^∞ manifold $\mathcal{M}'$ such that:*

- *$\mathcal{M}$ and $\mathcal{M}'$ are ε-close as embedded manifolds in the C^1 sense.*
- *The reach R' of $\mathcal{M}'$ satisfies $R' \geq R - \varepsilon$.*

▶ Remark 2. We can either remove the compactness assumption (because we only need a locally finite cover in the proof of the theorem) or we can assume that the reach is not decreased by ε. More precisely, if the manifold $\mathcal{M}$ is compact, we can increase the reach R' of the manifold $\mathcal{M}'$ by scaling it by a factor $R/(R-\varepsilon)$, and achieve $R' \geq R$. However, this may increase the distance between $\mathcal{M}$ and $\mathcal{M}'$ in the C^1 sense (by $\mathcal{O}(\varepsilon)$).

In addition, our result can be restated in terms of density in the space of submanifolds:

▶ **Corollary 3.** *The space of C^∞ embedded submanifolds of $\mathbb{R}^d$ with reach R is dense (in the C^1 topology) in the space of $C^{1,1}$ embedded submanifolds of $\mathbb{R}^d$ with reach R.*

This reformulation of the main theorem stands in a long line of density results from functional analysis and differential topology[2]. Such results have been widely used to extend results from a smaller to a larger class of functions, without any extra effort. Hence, our result establishes a new link between computational geometry, topology, and manifold learning on the one hand, and functional analysis and differential topology on the other.

The main results from this paper might appear trivial to experts in differential topology at first glance: It is well known that Lipschitz functions can be smoothed while preserving the Lipschitz bounds of both the function and its derivative – a fact that is straightforward and proved succinctly in Lemmas 15 and 16. However, this observation alone is far from sufficient to achieve the main result of the paper.

Outline. The structure of the paper closely follows the different steps in the proof of Theorem 1. Let us provide you with an outlook:
Step 1: We start with a compact submanifold $\mathcal{M}$ of $\mathbb{R}^d$ of positive reach. For each point $p \in \mathcal{M}$, we can find a neighbourhood in which $\mathcal{M}$ is a *graph of a function from the (affine) tangent to the normal space at p*. Since the reach of $\mathcal{M}$ is positive, this function is $C^{1,1}$, and we use the bound on the Lipschitz constant of its derivative to control the angles between nearby tangent spaces.

We fix an ε and select a sample of points $p \in \mathcal{M}$ whose neighbourhoods cover $\mathcal{M}$ (see Figure 4). We only work with one neighbourhood at a time.

[2] We refer to Chapter 2 of [24] for an extensive overview.

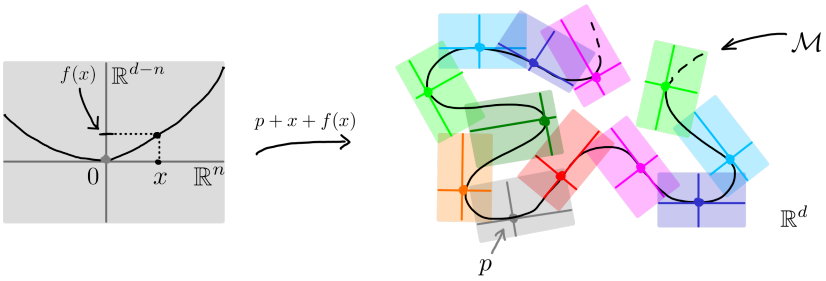


Figure 4 First we cover the manifold $\mathcal{M}$ with neighbourhoods in each of which $\mathcal{M}$ is representable as a graph of a function.

Step 2: We identify the point p with $0 \in \mathbb{R}^d$, and the tangent and normal spaces $T_p\mathcal{M}$ and $N_p\mathcal{M}$ with the first n and last $d-n$ coordinates of $\mathbb{R}^d$, respectively. Following this identification, we denote the map describing $\mathcal{M}$ by f, and its domain by U.

Our first goal is to smooth f in a neighbourhood of 0. We split U into three regions: a neighbourhood U_1 of 0, a region U_3 covering the vicinity of ∂U, and a transition region U_2 in between. We then use *kernel-based smoothing* to define a function

$$F : U \subseteq \mathbb{R}^n \to \mathbb{R}^{d-n}$$

that is smooth in U_1 and equals f in U_3 (see Figure 5). To achieve this, we employ a *partition of unity function*. We revise the background on smoothing and partitions of unity in Section 2.1. These techniques allow us to control the Lipschitz constant of F and its derivative, on which we give explicit bounds in Sections 3.1 and 3.2. We use *operator norms* to formulate these bounds. The background on operator norms is also presented in Section 2.1.

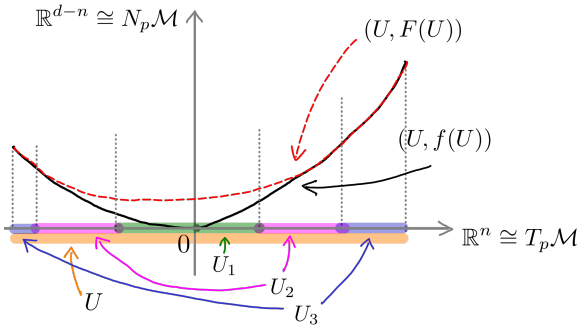


Figure 5 The function F (red) is smooth on the set U_1 (green) and equals f on the set U_3 (blue).

Step 3: We perform surgery on the manifold $\mathcal{M}$, and replace the graph of $p+f$ by the graph of $p+F$. We abuse notation and call this manifold $\mathcal{M}'$, although it is "only" smooth in a neighbourhood of the point p for now. Then we estimate the reach of $\mathcal{M}'$. To this end, we leverage a result by Federer, which characterizes the reach of a manifold through the distance from a point on it to the affine tangent space of another point on it. We pick two points in $p', q' \in \mathcal{M}'$ and investigate the distance from q' to the affine tangent space of p'. It turns out that bounding this distance is straightforward when p' does **not** lie in the graph of $p+F$. In the other case, we establish the bound using the relationship between the functions F and f, and the Lipschitz constants of f and its derivative. We cover these results in Section 3.3.

Step 4: We repeat Steps 2 and 3 iteratively for each point p of our sample, until we have smoothed the whole of $\mathcal{M}$. The process is illustrated in Figure 6. In each iteration, we have a one-parameter freedom in the choice of the smoothing kernel.

In this final step of the proof, we show that both the point sample and the smoothing kernels can be chosen in such a way that at the end, the smooth manifold $\mathcal{M}'$ satisfies the conditions of Theorem 1. This final step is described in Section 4.

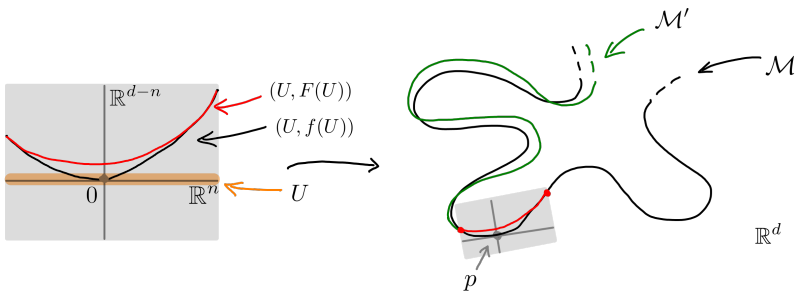


Figure 6 We construct the manifold $\mathcal{M}'$ iteratively. In each neighbourhood (gray), we replace the original manifold ($(U, f(U))$, in black) by a smooth piece ($(U, F(U))$, in red).

We indicate all the results in the paper that are (most probably) well known to experts in differential topology as "Folklore". We provide proofs of these statements,as well as the proofs of all our statements except the main theorem, in the full version of this article.

2 Preliminaries

We assume that the reader is reasonably familiar with sets of positive reach. Furthermore, we recall a characterization of the reach given by Federer [20] and results on Lipschitz-continuity of maps from a tangent space into a normal space of a manifold in the full version of this article.

2.1 Results from differential topology

In this section we recall three elementary tools from differential topology: partition of unity functions, the smoothing process, and operator norms. We adopt the formulation and notation used by Hirsch [24, Chapter 2].

Partition of unity functions

Partition of unity functions allow us to localize constructions and proofs in differential topology. They are defined as follows:

▶ **Definition 4.** *Let M be a C^k manifold, with $0 \leq k \leq \infty$, and $\mathcal{U} = \{U_i\}_{i \in I}$, with an index set I, an open cover of M. A* C^k partition of unity *subordinate to $\mathcal{U}$ is a family of C^k maps $\psi_i : M \to [0, 1]$, $i \in I$, with the following properties:*

- *For every $i \in I$, the support*[3] $\operatorname{supp}(\psi_i)$ *of ψ_i is contained in the set U_i.*
- *The collection $\{\operatorname{supp}(\psi_i)\}_{i \in I}$ of the supports of ψ_i is locally finite.*
- *The maps ψ_i sum to the function that is identically equal to 1, that is,*

$$\sum_i \psi_i(x) = 1.$$

[3] The support of a function ψ is the closure of the set $\psi^{-1}(\mathbb{R} \setminus \{0\})$.

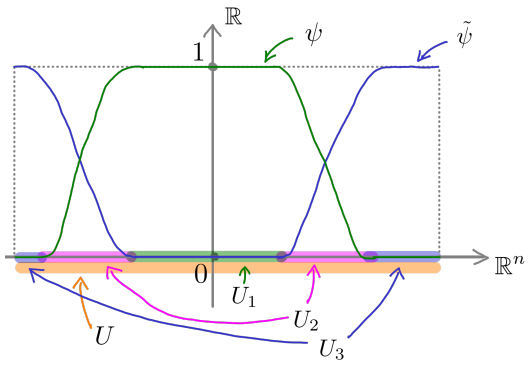


Figure 7 A cover $\{U_1 \cup U_2, U_2 \cup U_3\}$ of a set $U \subseteq \mathbb{R}^n$, and the corresponding family $\{\psi, \tilde{\psi}\}$ of partition of unity functions.

We illustrate the concept for our setting in Figure 7. We can find a partition of unity for any open cover:

▶ **Theorem 5** (Theorem 2.1 of [24]). *Let M be a C^k manifold with $1 \leq k \leq \infty$. Every open cover of M has a subordinate C^k partition of unity.*

Kernel smoothing

Kernel smoothing is, broadly speaking, a process that averages a map using a *kernel* function, making the result at least as smooth as the kernel itself. The kernel is defined as follows:

▶ **Definition 6** (Smoothing kernel). *A (smooth) map $\varphi : \mathbb{R}^n \to \mathbb{R}$ is called a* convolution *or a* smoothing kernel *if it is non-negative, has compact support, and $\int_{\mathbb{R}^n} \varphi = 1$.*

The support radius *of the smoothing kernel φ is the smallest value $\sigma \in \mathbb{R}_{\geq 0}$, for which the support* $\mathrm{supp}(\varphi)$ *of φ is contained in the closed ball of radius σ centred at the origin:*

$$\mathrm{supp}(\varphi) \subset B(0, \sigma) \subset \mathbb{R}^n.$$

We illustrate the smoothing radius in Figure 8.

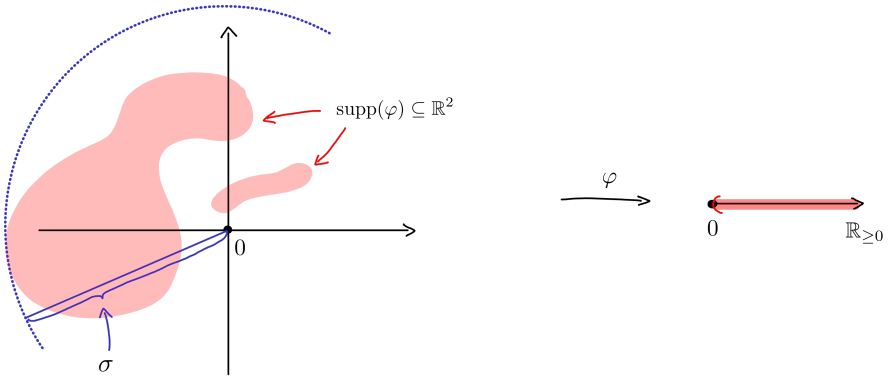


Figure 8 The support radius of the smoothing kernel φ.

Smoothing relies on neighbourhoods, determined by the support radius of the kernel. For smoothing of a map on a given set to be well-defined, the map itself must be well-defined on a sufficiently thick neighborhood surrounding the set. Consequently, it is sometimes necessary to shrink the domain of the map where the smoothing will be applied:

▶ **Definition 7.** *Consider a smoothing kernel with support radius σ. Given an open set $U \subset \mathbb{R}^n$, the* shrinking U_σ *of U is defined as*

$$U_\sigma = \{x \in U | B(x, \sigma) \subset U\}.$$

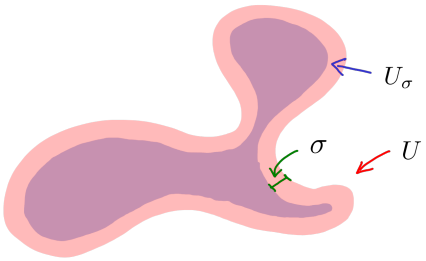


Figure 9 The shrinking of a set as defined in Definition 7.

The smoothing process, or in other words, the *convolution*, is carried out through integration:

▶ **Definition 8** (Convolution)**.** *Let $\varphi : \mathbb{R}^n \to \mathbb{R}$ be a smoothing kernel with support radius σ, $U \subset \mathbb{R}^n$ an open set, and $f : U \to \mathbb{R}^{d-n}$ a continuous map. The* convolution *of f by φ is the map*

$$\varphi * f : U_\sigma \to \mathbb{R}^{d-n}, \qquad x \mapsto \varphi * f(x) = \int_{\mathbb{R}^n} \varphi(y) f(x-y) \mathrm{d}y.$$

Smoothing improves the smoothness of the map, and commutes with differentiation:

▶ **Theorem 9** ([24, Theorem 2.3 (a) and (b)])**.** *Let $\varphi : \mathbb{R}^n \to \mathbb{R}$ be a smoothing kernel with support radius $\sigma > 0$, $U \subset \mathbb{R}^n$ an open set and $f : U \to \mathbb{R}^{d-n}$ a continuous map. The convolution $\varphi * f : U_\sigma \to \mathbb{R}^{d-n}$ has the following properties:*

- *If φ is C^k, with $1 \leq k \leq \infty$, then so is $\varphi * f$, and for each finite $\ell \leq k$,*

$$D^\ell(\varphi * f) = D^\ell(\varphi) * f$$

  on U_σ.
- *If f is C^k, with $1 \leq k \leq \infty$, then so is $\varphi * f$, and for each finite $\ell \leq k$,*

$$D^\ell(\varphi * f) = \varphi * (D^\ell f).$$

Operator norms

The last result we need is the convergence of smoothed maps to the original map. We consider convergence of the map itself as well as its first k derivatives with respect to the so-called C^k *norm*. We define this norm in two steps:

▶ **Definition 10.** *The norm $\|.\|$ of a k-linear map*

$$S : \mathbb{R}^n \times \cdots \times \mathbb{R}^n \to \mathbb{R}^m, \qquad (u_1, \ldots, u_k) \mapsto S(u_1, \ldots, u_k),$$

is defined as

$$\|S\| = \max_{|u_i|=1} |S(u_1, \ldots, u_k)|.$$

Consider a C^k map $f : U \to \mathbb{R}^{d-n}$, and its rth order derivative $D^r f$. For $r \geq 1$, the map $D^r f(x)$ is an r-linear map at each point $x \in U$, and we use the above definition to measure the norm $\|D^r f(x)\|$ of $D^r f$ at x. For $r = 0$, the notation $\|D^r f(x)\| = \|D^0 f(x)\|$ should be understood as $|f(x)|$.

▶ **Definition 11** (C^k norm). *Let $f : U \to \mathbb{R}^{d-n}$ be a C^k map, $U \subset \mathbb{R}^n$ an open set, and $C \subset U$ any subset of U. The C^k norm of f on C is defined as*

$$\|f\|_{k,C} = \sup\{\|D^r f(x)\| \mid x \in C, 0 \leq r \leq k\}.$$

We can now define the convergence we will use:

▶ **Theorem 12** ([24, Theorem 2.3 (c)]). *Let $U \subset \mathbb{R}^n$ be an open set with a compact subset $C \subset U$, and $f : U \to \mathbb{R}^{d-n}$ a C^k map, with $0 \leq k \leq \infty$. For any $\varepsilon > 0$ there exists a value $\sigma > 0$ such that $C \subset U_\sigma$, and any C^k smoothing kernel φ with support radius σ satisfies*

$$\|\varphi * f - f\|_{k,C} \leq \varepsilon.$$

▶ Remark 13. The operator 2-norm of a matrix A is defined as

$$\|A\|_2 = \sup_{v \neq 0} \frac{|Av|}{|v|} = \max_{|v|=1} |Av|,$$

where $|\cdot|$ denotes the usual Euclidean (2-)norm.

For the first derivative of a function f, evaluated at a point x, the norm from Definition 10 coincides with the operator 2-norm, that is, $\|D^1 f(x)\| = \|D^1 f(x)\|_2$. In particular,

$$\|Df\|_{0,C} = \sup\{\|Df(x)\|_2 \mid x \in C\} \qquad \text{and} \qquad \|f\|_{1,C} = \max\{\|f\|_{0,C}, \|Df\|_{0,C}\}.$$

In this paper, we primarily focus on the case where $k = 1$, as estimating the first derivative proves to be the main challenge in our proofs. Consequently, much of our work involves dealing with operator 2-norms.

In addition, we often make a choice of the map f on which we perform the smoothing. The lemma below implies that our choice does not depend on the map f itself, but only on its Lipschitz constant.

▶ **Lemma 14** (Folklore). *We adapt the settings from Theorem 12. If the map f is L-Lipschitz, then for any point $x \in U_\sigma$ and $y \in B(0, \sigma)$ it holds that $|f(x - y) - f(x)| \leq L\sigma$. As a consequence,*

$$\|\varphi * f - f\|_{0,C} \leq L\sigma.$$

3 On the control over Lipschitz constants while smoothing and using partition of unity functions

This section consists of three parts, in which we

- recall that smoothing by convolution does not affect Lipschitz constants;
- set the stage to define the function F that locally describes our smoothed manifold $\mathcal{M}'$, and determine the Lipschitz constant of F and its derivative;
- establish two bounds on the distance between the affine tangent space of a point on the graph of F, and another point in $\mathcal{M}'$.

3.1 Lipschitz constants for smoothings by convolution

In this section we focus on Lipschitz constants. To this end, we denote the Lipschitz constant of a function g by L_g. We recall that a function $g : \mathbb{R}^n \to \mathbb{R}^m$ is Lipschitz with constant L_g if for all points $y_1, y_2 \in \mathbb{R}^n$,

$$|g(y_2) - g(y_1)| \leq L_g \, |y_2 - y_1| \,. \tag{1}$$

Similarly, if g is differentiable, the derivative Dg of g is Lipschitz with constant L_{Dg} if for all points $y_1, y_2 \in \mathbb{R}^n$,

$$\left\|Dg(y_2) - Dg(y_1)\right\|_2 \leq L_{Dg}|y_2 - y_1|. \tag{2}$$

We first recall that smoothing does not influence the Lipschitz constant of a function or its derivative.

The result of the convolution of a function g by a kernel is just a barycenter of translates of g. Since the minimal Lipschitz constant satisfied by a function defines a semi-norm on functions, it is a convex functional. As a result:

▶ **Lemma 15** (Folklore). *Let $g : \mathbb{R}^n \to \mathbb{R}^{d-n}$ be an L_g-Lipschitz map and $\varphi : \mathbb{R}^n \to \mathbb{R}$ a smoothing kernel. Then the smoothing by convolution $\varphi * g$ is also L_g-Lipschitz.*

Moreover, since convolution commutes with derivation, we get:

▶ **Lemma 16** (Folklore). *Let $g : \mathbb{R}^n \to \mathbb{R}^{d-n}$ be a function whose* first derivative *is L_{Dg}-Lipschitz. Let further $\varphi : \mathbb{R}^n \to \mathbb{R}$ be a smoothing kernel. Then the first derivative of the smoothing by convolution $\varphi * g$ is also L_{Dg}-Lipschitz.*

3.2 Interpolation between a Lipschitz function and its smoothing by convolution

In this section, we consider an interpolation F between a map f locally describing our manifold, and its convolution $\varphi * f$, using a partition of unity function ψ. A formal definition of the function F will follow shortly. We illustrate this construction in Figure 10.

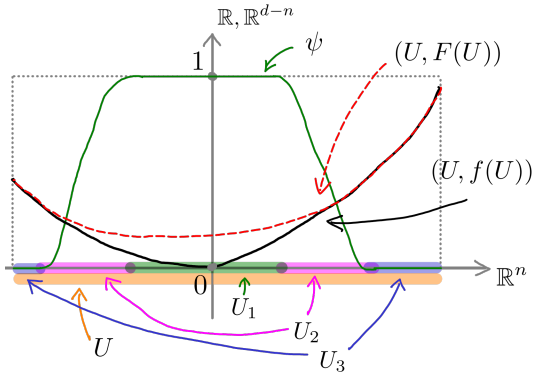


Figure 10 The superposed graphs of f, F, and ψ show that F is smooth on U_1 and $F = f$ on U_3.

To be able to use local arguments, we need to prove that the Lipschitz constants of F and its first derivative are close to the Lipschitz constants of f and its first derivative.

The Lipschitz constants of the function F were studied in [9], leading to a result similar to our Lemma 17. Our key contribution lies in determining the Lipschitz constant of the *derivative* of F, a significantly more intricate task and one that is crucial for proving the main result.

Throughout the rest of this paper, we operate under the following settings:

- We consider a function $f : U \subseteq \mathbb{R}^n \to \mathbb{R}^{d-n}$ whose graph is (locally) the manifold $\mathcal{M}$. Such a map exists due to Theorem 27 of the full version of this article. Moreover, both f and its derivative Df are Lipschitz, and we denote their Lipschitz constants by L_f and L_{Df}, respectively.
- We cover the domain U with two sets $\{U_1 \cup U_2, U_2 \cup U_3\}$ and consider a partition of unity subordinate to this cover. We denote the partition of unity function corresponding to the set $U_1 \cup U_2$ by ψ, and the closure of its support by

  $$\overline{\operatorname{supp} \psi} = C.$$

  We assume C to be compact. We denote the Lipschitz constants of ψ and its derivative $D\psi$ by L_ψ and $L_{D\psi}$, respectively, write

  $$L_{\psi,D\psi} = \max\{L_\psi, L_{D\psi}\}, \tag{3}$$

  and note that

  $$\sup_{y\in C} \|D\psi(y)\|_2 = \|D\psi\|_{0,C} \leq L_\psi \leq L_{\psi,D\psi}. \tag{4}$$

- For $\varepsilon > 0$ we choose a smoothing kernel $\varphi_\varepsilon : U \to \mathbb{R}$ such that

  $$\|\varphi_\varepsilon * f - f\|_{1,C} \leq \varepsilon. \tag{5}$$

  Such kernel exists due to Theorem 12. We recall (see Remark 13) that inequality (5) holds if and only if both of the following inequalities hold:

  $$\sup_{y\in C} \|(\varphi_\varepsilon * Df - Df)(y)\|_2 = \|\varphi_\varepsilon * Df - Df\|_{0,C} \leq \varepsilon, \tag{6}$$

  and

  $$\sup_{y\in C} |(\varphi_\varepsilon * f - f)(y)| = \|\varphi_\varepsilon * f - f\|_{0,C} \leq \varepsilon. \tag{7}$$

- The (locally) smoothed manifold $\mathcal{M}'$ is then locally described by the graph of the function

  $$F : U \to \mathbb{R}^{d-n}, \qquad x \mapsto F(x) = (1 - \psi(x))f(x) + \psi(x)(\varphi_\varepsilon * f(x)). \tag{8}$$

▶ **Lemma 17.** *The function F is Lipschitz on C with Lipschitz constant $(L_\psi\varepsilon + L_f)$.*

▶ **Lemma 18.** *The first derivative DF of the function F is Lipschitz on C with Lipschitz constant $(3L_{\psi,D\psi}\varepsilon + L_{Df})$.*

3.3 On bounding the reach of $\mathcal{M}'$ in terms of Lipschitz constants

Let $\mathcal{M}'$ denote the manifold that

- equals the graph of the function $p + F$ inside the neighbourhood of the point p;
- equals $\mathcal{M}$ outside of this neighbourhood.

Our next goal is to bound the reach of $\mathcal{M}'$. To this end, we use a result by Federer [20] that characterizes the reach of a manifold through the distance from a point on it to a tangent space of another point on it. We provide two different bounds on this distance. In order to prove the latter, we also establish bounds on the angle between tangent spaces of the graphs of f and F.

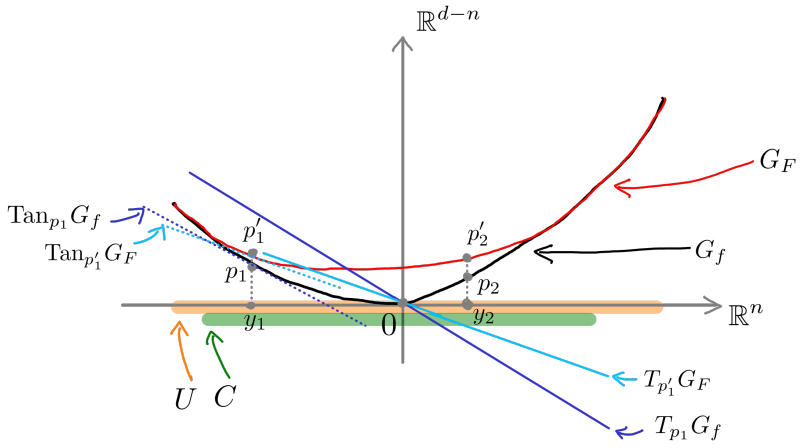


Figure 11 Illustration of the setup for this section.

We adapt the following settings, in addition to the ones established in Section 3.2: We fix two points $y_1, y_2 \in C, y_1 \neq y_2$, and label their graphs by

$$p_1 = (y_1, f(y_1)), \quad p_2 = (y_2, f(y_2)) \qquad \text{and} \qquad p_1' = (y_1, F(y_1)), \quad p_2' = (y_2, F(y_2)).$$

We write G_f and G_F for the graph of f and F, respectively, and $T_{p_1} G_f$ and $\mathrm{Tan}_{p_1} G_f$ (resp. $T_{p_1'} G_F$ and $\mathrm{Tan}_{p_1'} G_F$) for the tangent and affine tangent space of f at p_1 (resp. of F at p_1').

We illustrate this setup in Figure 11.

▶ **Lemma 19.** *The distance between the point p_2' and the affine tangent space $\mathrm{Tan}_{p_1'} G_F$ is bounded by*

$$d(p_2', \mathrm{Tan}_{p_1'} G_F) \leq \tfrac{1}{2}(3L_{\psi, D\psi}\varepsilon + L_{Df}) \left|p_2' - p_1'\right|^2. \tag{9}$$

This has the following consequence: Let $\rho > 0$ be small enough that the ρ-neighbourhood of y_1 is contained in C, $B(y_1, \rho) \subseteq C$. Then the graph G_F of the function F in this neighbourhood is contained in the union of balls

$$\bigcup_{v \in \mathbb{R}^n : |v| = r \leq \rho} B\left(p_1' + \begin{pmatrix} v \\ DF(y_1)v \end{pmatrix}, \tfrac{1}{2}(3L_{\psi, D\psi}\varepsilon + L_{Df})\, r^2\right).$$

We illustrate the settings of Lemma 19 in Figure 12.

As an auxiliary result needed to prove Proposition 22, we obtain a bound on the angle between the affine tangent spaces $\mathrm{Tan}_{p_1} G_f$ and $\mathrm{Tan}_{p_1'} G_F$, as well as a bound on the Hausdorff distance between two neighbourhoods contained in these spaces. We note that the angle between two affine spaces is equal to the angle between the corresponding vector spaces. The angle between two vector subspaces A and B is defined as

$$\angle A, B \underset{\text{def.}}{=} \max_{a \in A \setminus \{0\}} \min_{b \in B \setminus \{0\}} \angle a, b = \max_{b \in B \setminus \{0\}} \min_{a \in A \setminus \{0\}} \angle a, b.$$

▶ **Lemma 20.** *Assume that $\varepsilon \leq \frac{1}{L_{\psi, D\psi} + 1}$. Then the angle between the affine tangent spaces $\mathrm{Tan}_{p_1} G_f$ and $\mathrm{Tan}_{p_1'} G_F$ is bounded by*

$$\angle(\mathrm{Tan}_{p_1} G_f, \mathrm{Tan}_{p_1'} G_F) \leq \arcsin\left(L_{\psi, D\psi}\varepsilon + \varepsilon\right).$$

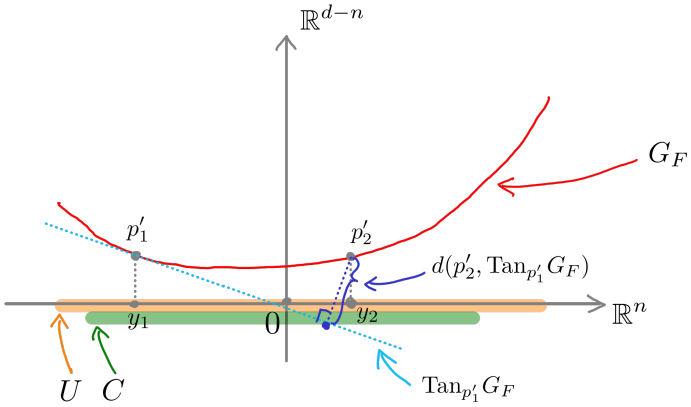


Figure 12 Illustration of the settings of Lemma 19.

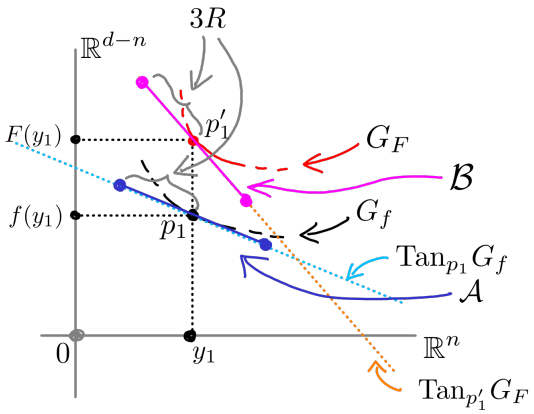


Figure 13 The $3R$-neighbourhoods of the points p_1 (in dark blue) and p'_1 (in pink) in the corresponding affine tangent spaces.

▶ **Corollary 21.** *Let $R > 0$ denote the reach of the manifold $\mathcal{M}$, and consider the n-dimensional neighbourhood of size $3R$ of the point p_1 in the affine tangent space $\mathrm{Tan}_{p_1} G_f$:*

$$\mathcal{A} = p_1 + \{T \in T_{p_1} G_f \mid |T| \leq 3R\}.$$

Similarly, consider a $3R$-neighbourhood of the point p'_1 in the affine tangent space $\mathrm{Tan}_{p'_1} G_F$:

$$\mathcal{B} = p'_1 + \{T \in T_{p'_1} G_F \mid |T| \leq 3R\}.$$

Assume that $\varepsilon \leq \frac{1}{L_{\psi, D\psi}+1}$. Then the Hausdorff distance between the two neighbourhoods is upper bounded:

$$d_H(\mathcal{A}, \mathcal{B}) \leq \varepsilon \left(6RL_{\psi, D\psi} + 6R + 1\right).$$

The settings of Corollary 21 are illustrated in Figure 13.

At last, we bound the distance between the affine tangent space $p + \mathrm{Tan}_{p'_1} G_F$ and a point $q' \in \mathcal{M}'$ that does not lie on the graph of $p + F$. We illustrate the settings in Figure 14.

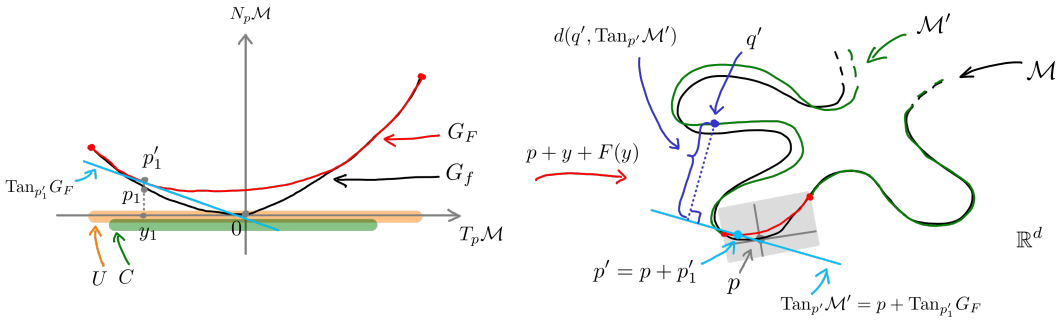


Figure 14 Illustration of the setup of Proposition 22.

▶ **Proposition 22.** *Let $q' \in \mathcal{M}' \backslash (p + G_F)$, and assume that $\varepsilon \leq \frac{1}{L_{\psi, D\psi}+1}$. Write $p + p_1' = p'$. Then the distance between the point q' and the affine tangent space $\mathrm{Tan}_{p'}\,\mathcal{M}' = p + \mathrm{Tan}_{p_1'}\,G_F$ is bounded by*

$$d(q', \mathrm{Tan}_{p'}\,\mathcal{M}') \leq \frac{|q' - p'|^2}{2R} + \frac{\varepsilon^2}{2R} + \varepsilon\,(6RL_{\psi, D\psi} + 6R + 4)\,. \tag{10}$$

Therefore, if $\varepsilon < R$ and $|q' - p'| \geq \beta$, then

$$d(q', \mathrm{Tan}_{p'}\,\mathcal{M}') \leq \frac{|q' - p'|^2}{2R'},$$

with

$$R' = \frac{R}{1 + \frac{12\varepsilon R}{\beta^2}\,(RL_{\psi, D\psi} + R + 1)}.$$

4 Proof of the main theorem

We now have all the necessary tools to prove our main result. Before proceeding, let us restate the theorem:

▶ **Theorem 1.** *Let $\mathcal{M} \subset \mathbb{R}^d$ be a compact manifold of (positive) reach R, and $\varepsilon > 0$. Then there exists a C^∞ manifold $\mathcal{M}'$ such that:*

- *$\mathcal{M}$ and $\mathcal{M}'$ are ε-close as embedded manifolds in the C^1 sense.*
- *The reach R' of $\mathcal{M}'$ satisfies $R' \geq R - \varepsilon$.*

Proof. We proceed as discussed in the introduction.

Step 1. In the first step we select a sample P of the manifold $\mathcal{M}$ (see also Figure 15, right). To this end, we choose a parameter $0 < \delta \leq R/2$, and let P be a $\sqrt{\delta R}/16$-net[4] on $\mathcal{M}$, meaning that

- for every point $x \in \mathcal{M}$ one can find a point $\mathsf{p} \in \mathsf{P}$, such that $|x - \mathsf{p}| \leq \sqrt{\delta R}/16$,
- for all $\mathsf{p}, \mathsf{q} \in \mathsf{P}$, $|\mathsf{p} - \mathsf{q}| \geq \sqrt{\delta R}/16$.

Such a net exists thanks to [13, Lemma 5.2, Section 5.1.1].

[4] The factor 16 in $\sqrt{\delta R}/16$ is almost certainly suboptimal. However, it simplifies the proof by a margin.

Due to these properties, the $\sqrt{\delta R}/16$-balls centred at the points of the sample P cover $\mathcal{M}$. In addition, we consider balls of radius $\sqrt{\delta R}/2$ centred at the points of P.

We need one more ingredient before we start iteratively visiting neighbourhoods of each point $\mathsf{p} \in \mathsf{P}$ – a fixed partition of unity function, which we rescale at every iteration. To this end, we split the ball $B(0,4) \subseteq \mathbb{R}^n$ into three sets,

$$V_1 = B(0,1), \quad V_2 = \{x \in \mathbb{R}^n \mid 1 \leq |x| < 2\}, \quad V_3 = \{x \in \mathbb{R}^n \mid 2 \leq |x| \leq 4\}.$$

The two sets $\{V_1 \cup V_2, V_2 \cup V_3\}$ then cover the ball $B(0,4)$, and we let ψ_0 be the partition of unity function corresponding to the set $V_1 \cup V_2$. Then $\psi_0 \equiv 1$ on V_1, and $\psi_0 \equiv 0$ on V_3.

Next, we select a point $\mathsf{p} \in \mathsf{P}$, and restrict our attention to the ball $B(\mathsf{p}, \sqrt{\delta R}/2)$ of radius $\sqrt{\delta R}/2$ centred at p.

Step 2. Let f be a function whose graph describes the manifold $\mathcal{M}$ in the ball $B(\mathsf{p}, \sqrt{\delta R}/2)$. To be more concrete, choose $f : T_{\mathsf{p}}\mathcal{M} \cap B(0, \sqrt{\delta R}/2) \to N_{\mathsf{p}}\mathcal{M}$ as in Theorem 27 in the full version of this article, or [26] which gives a more extensive version of Theorem 27. Due to this theorem and our choice of δ, the derivative of f is $\frac{1}{R-\delta}$-Lipschitz on the whole domain $T_{\mathsf{p}}\mathcal{M} \cap B(0, \sqrt{\delta R}/2)$.

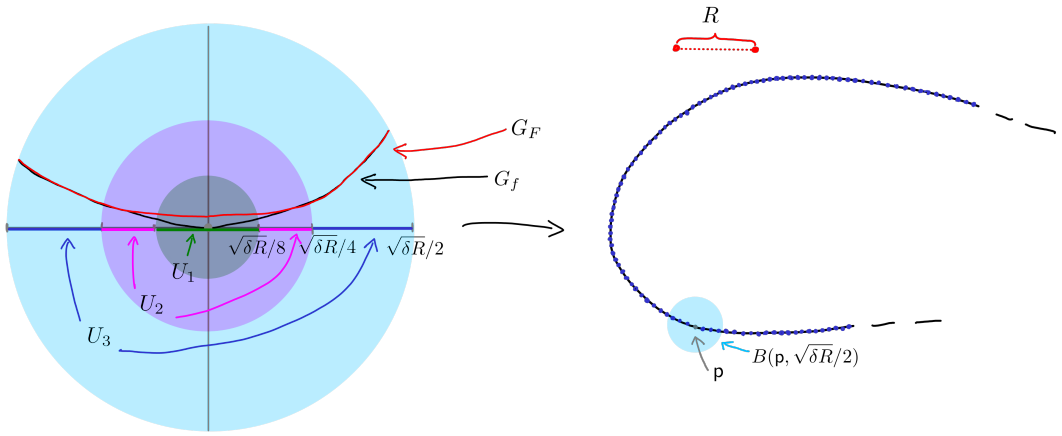


Figure 15 The point sample P at a piece of the manifold $\mathcal{M}$, and the neighbourhoods U_1, U_2, and U_3.

We apply the smoothing construction described in Section 3.2. As ingredients, we need a partition of unity function, and a smoothing kernel.

Partition of unity function: We map the neighbourhood $T_{\mathsf{p}}\mathcal{M} \cap B(0, \sqrt{\delta R}/2)$ diffeomorphically to the ball $B(0,4) \subseteq \mathbb{R}^n$ using the canonical identification of the tangent space $T_{\mathsf{p}}\mathcal{M}$ with $\mathbb{R}^n$ and the map $x \mapsto \frac{8}{\sqrt{\delta R}} \cdot x$. The preimages of the sets V_1, V_2, and V_3 under this map are, respectively,

$$U_1 = \left\{x \in T_{\mathsf{p}}\mathcal{M} \mid |x| < \sqrt{\delta R}/8\right\},$$
$$U_2 = \left\{x \in T_{\mathsf{p}}\mathcal{M} \mid \sqrt{\delta R}/8 \leq |x| < \sqrt{\delta R}/4\right\},$$
$$U_3 = \left\{x \in T_{\mathsf{p}}\mathcal{M} \mid \sqrt{\delta R}/4 \leq |x| \leq \sqrt{\delta R}/2\right\}.$$

The sets U_1, U_2, and U_3 are illustrated in Figure 15, on the left. We define ψ by scaling the partition of unity function ψ_0: $\psi(x) = \psi_0\left(\frac{8x}{\sqrt{\delta R}}\right)$. Then ψ is the partition of unity function corresponding to the set $U_1 \cup U_2$ in the cover $\{U_1 \cup U_2, U_2 \cup U_3\}$ of $T_{\mathsf{p}}\mathcal{M} \cap B(0, \sqrt{\delta R}/2)$, and thereby, $\psi \equiv 1$ on U_1, and $\psi \equiv 0$ on U_3. Furthermore:

▶ **Lemma 23.** *The Lipschitz constants of ψ and its derivative are bounded by $L_\psi \leq \frac{8}{\sqrt{\delta R}} L_{\psi_0}$ and $L_{D\psi} \leq \frac{64}{\delta R} L_{D\psi_0}$.*

By decreasing δ if necessary, we can also assume that $\frac{8}{\sqrt{\delta R}} L_{\psi_0} \leq \frac{64}{\delta R} L_{D\psi_0}$. Thus:

$$L_{\psi, D\psi} = \max\{L_\psi, L_{D\psi}\} \leq \frac{64}{\delta R} L_{D\psi_0}. \tag{11}$$

Smoothing kernel: Let $C = \overline{\operatorname{supp} \psi}$. For $0 < \rho < \frac{1}{1+\frac{64}{\delta R} L_{D\psi_0}}$ we choose a smoothing kernel $\varphi_\rho : T_{\mathsf{p}}\mathcal{M} \cap B(0, \sqrt{\delta R}/2) \to \mathbb{R}$ such that

$$\|\varphi_\rho * f - f\|_{1,C} \leq \rho. \tag{5}$$

Such kernel exists due to Theorem 12. With these choices we define

$$F : T_{\mathsf{p}}\mathcal{M} \cap B(0, \sqrt{\delta R}/2) \to N_{\mathsf{p}}\mathcal{M}, \quad x \mapsto F(x) = (1 - \psi(x))f(x) + \psi(x)(\varphi_\rho * f(x)). \tag{8}$$

Step 3. We write $\mathcal{M}'$ for the manifold that coincides with $\mathcal{M}$ outside of the ball of radius $\sqrt{\delta R}/2$ centred at the point p, and is the graph of $\mathsf{p} + F$ inside this ball.

To be more concrete, we recall that we can write each point $p \in \mathcal{M} \cap B(\mathsf{p}, \sqrt{\delta R}/2)$ as $p = \mathsf{p} + y + f(y)$, with $y \in T_{\mathsf{p}}\mathcal{M}$. With this in mind we define, for each $p \in \mathcal{M}$:

$$p' \underset{\text{def.}}{=} \begin{cases} \mathsf{p} + y + F(y), & \text{if } p \in B(\mathsf{p}, \sqrt{\delta R}/2), \\ p, & \text{otherwise,} \end{cases}$$

and let $\mathcal{M}' = \{p' \mid p \in \mathcal{M}\}$.

We stress that

- inside the ball $B(\mathsf{p}, \sqrt{\delta R}/16)$, $\mathcal{M}'$ is smooth (C^∞), and
- $\mathcal{M}' = \mathcal{M}$ not only outside of the ball $B(\mathsf{p}, \sqrt{\delta R}/2)$, but already outside of the ball $B(\mathsf{p}, \sqrt{\delta R}/4)$.

Next, we set forth to bound the reach of $\mathcal{M}'$. We choose two points $p', q' \in \mathcal{M}'$, and estimate the distance between q' and the affine tangent space $\operatorname{Tan}_{p'} \mathcal{M}'$ – our bounds from Lemma 19 and Proposition 22 yield:

▶ **Lemma 24.** *Let $p', q' \in \mathcal{M}'$ be two points in the manifold $\mathcal{M}'$. Then the distance between q' and the affine tangent space $\operatorname{Tan}_{p'} \mathcal{M}'$ is bounded by*

$$d(q', \operatorname{Tan}_{p'} \mathcal{M}') \leq \frac{|p' - q'|^2}{2R'},$$

with

$$R' = \frac{R}{\max\left\{3 \cdot 64 L_{D\psi_0} \frac{\rho}{\delta} + \frac{1}{1 - \frac{\delta}{R}}, 1 + 3 \cdot 64 \frac{\rho}{\delta}\left(\frac{64}{\delta} L_{D\psi_0} + R + 1\right)\right\}}. \tag{12}$$

Thus, due to Federer's Theorem [20, Theorem 4.18], the reach of the manifold $\mathcal{M}'$ satisfies $\operatorname{rch}(\mathcal{M}') \geq R'$, with R' as in equation (12).

Step 4. Here we apply the smoothing construction iteratively for all points of P. The correctness of the construction relies on two observations: The first observation is that (12) can be made as close to R as needed. The second observation is that the manifold is locally changed only a finite number of times that (which in the non-compact case requires specific choices of δ, ϵ, and ρ).

Because P is a $\sqrt{\delta R}/16$-net, it is locally finite. In particular, the number of points of P contained in a ball of radius $\sqrt{\delta R}/2$ centred at any point is $\mathcal{O}(8^d)$, as can be seen by a simple packing argument, see e.g. [13, Lemma 5.3, Section 5.1.1] for similar arguments. However, with every iteration of the construction, the points in a $\sqrt{\delta R}/2$ neighbourhood of the point p (which is the centre of the local smoothing operation) move[5] by at most ρ, due to (5).

This means that the points can move at most by $N_V\rho$, where N_V is the maximal number of times any point is visited. We claim that

▷ Claim 25. If we pick $N_C = 8^{d+1}$ and assume that $\rho \leq \frac{\sqrt{\delta R}/16}{2\cdot 8^{d+1}} = \frac{\sqrt{\delta R}/32}{N_C}$, then $N_V \leq N_C$.

Proof of Claim 25. Write $N_V(i)$ for the maximal number of times any point is visited after i iterations of the surgery process. The number $N_V(i)$ is upper bounded by the number of points of P that, during the iterative process (up to iteration i), end up in a ball of radius $\sqrt{\delta R}/2$ around the displaced point, which can move as well. Because the points move by at most $\rho N_V(i)$ and need to enter the ball of radius $\sqrt{\delta R}/2$, they must initially lie inside a ball of radius $\sqrt{\delta R}/2 + 2\rho N_V(i)$. In other words, in order for a surgery centered at point q to impact point p, the original positions of p and q must lie at distance at most $\sqrt{\delta R}/2 + 2\rho N_V(i)$ apart. Because the points in the sample are originally $\sqrt{\delta R}/16$ separated, a packing argument gives the following relation:

$$N_V(i) \leq \left(\frac{\sqrt{\delta R}/2 + 2\rho N_V(i)}{\sqrt{\delta R}/16} \right)^d . \tag{13}$$

If we take ρ sufficiently small, for example $\rho \leq \frac{\sqrt{\delta R}/16}{2\cdot 8^{d+1}}$, then equation (13) is not satisfied for $N_V(i) = N_C$ with $N_C = 8^{d+1}$. This means that for any i, we have that $N_V(i) \leq N_C$. In particular at the end of the process the maximal number of times any point is visited is upper bounded by N_C, i.e. $N_V \leq N_C$. ◁

For the given $\varepsilon > 0$, we now choose ρ and δ such that R' from equation (12) satisfies $|R - R'| \leq \varepsilon/(4N_C)$. This is always possible. One can set $\delta = \varepsilon/(4N_C)$; the choice for ρ is a bit more subtle but choosing $\rho = \mathcal{O}(\delta^4)$ works if δ (and equivalently ε) is sufficiently small. Additionally, as we have seen, δ has to be sufficiently small so that $\rho N_C = 8^{d+1}\mathcal{O}(\delta^4) \leq \sqrt{\delta R}/32$. This in particular means that even after the iterative process (that moves the points of P) the resulting set is still a $\sqrt{\delta R}/8$ cover, which means that every point in the manifold is smoothed. This is the reason we started with a $\sqrt{\delta R}/16$-net and U_1 is of size $\sqrt{\delta R}/8$.

Let $p, q \in \mathcal{M}$ be arbitrary. We now note that:

- Every pair of points p and q in the manifold is visited/perturbed by the smoothing process at most N_C times (each).
- We have picked ρ and δ such that $|R - R'| \leq \varepsilon/(4N_C)$, where here the difference between the reach R and R' is interpreted as between each iteration of the smoothing process.

[5] This $\sqrt{\delta R}/2$ size bound is not tight, in fact it can easily be improved to $\sqrt{\delta R}/4 + (\sqrt{\delta R}/4)^2$ by noting that the part of $\mathcal{M}$ whose projection onto $T_{\mathsf{p}}\mathcal{M} \cap B(0, \sqrt{\delta R}/2)$ falls outside U_2 is not perturbed.

This means that the characterization of the reach (for the points p and q) of Lemma 24 only changes $2N_C$ times, and thus for a total of less than ε. Because the choice of p and q is arbitrary, the reach after the smoothing has decreased by at most ε. ◀

References

1 Eddie Aamari, Clément Berenfeld, and Clément Levrard. Optimal reach estimation and metric learning. *The Annals of Statistics*, 51(3):1086–1108, 2023. doi:10.1214/23-AOS2281.

2 Eddie Aamari, Jisu Kim, Frédéric Chazal, Bertrand Michel, Alessandro Rinaldo, and Larry Wasserman. Estimating the reach of a manifold. *Electronic journal of statistics*, 13(1):1359–1399, 2019. doi:10.1214/19-EJS1551.

3 Eddie Aamari and Clément Levrard. Stability and minimax optimality of tangential Delaunay complexes for manifold reconstruction. *Discrete & Computational Geometry*, 59(4):923–971, 2018. doi:10.1007/s00454-017-9962-z.

4 Eddie Aamari and Clément Levrard. Nonasymptotic rates for manifold, tangent space and curvature estimation. *The Annals of Statistics*, 47(1), February 2019. doi:10.1214/18-aos1685.

5 Yariv Aizenbud and Barak Sober. Estimation of local geometric structure on manifolds from noisy data. *Journal of Machine Learning Research*, 26(64):1–89, 2025. URL: http://jmlr.org/papers/v26/25-0183.html.

6 Nina Amenta and Marshall Bern. Surface reconstruction by Voronoi filtering. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, SoCG '98, pages 39–48, New York, NY, USA, 1998. Association for Computing Machinery. doi:10.1145/276884.276889.

7 Nina Amenta and Marshall Bern. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, December 1999. doi:10.1007/PL00009475.

8 Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 415–421, New York, NY, USA, 1998. Association for Computing Machinery. doi:10.1145/280814.280947.

9 D. Azagra, J. Ferrera, F. López-Mesas, and Y. Rangel. Smooth approximation of Lipschitz functions on Riemannian manifolds. *Journal of Mathematical Analysis and Applications*, 326(2):1370–1378, 2007. doi:10.1016/j.jmaa.2006.03.088.

10 Mikhail Belkin, Jian Sun, and Yusu Wang. Discrete laplace operator on meshed surfaces. In *Proceedings of the Twenty-Fourth Annual Symposium on Computational Geometry*, SCG '08, pages 278–287, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1377676.1377725.

11 Clément Berenfeld, John Harvey, Marc Hoffmann, and Krishnan Shankar. Estimating the reach of a manifold via its convexity defect function. *Discrete & Computational Geometry*, 67(2):403–438, June 2021. doi:10.1007/s00454-021-00290-8.

12 Harry Blum. *A transformation for extracting new descriptors of shape*, volume 4. MIT press Cambridge, 1967.

13 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and Topological Inference*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2018. doi:10.1017/9781108297806.

14 S.-W. Cheng, T.K. Dey, and J.R. Shewchuk. *Delaunay Mesh Generation*. Computer and information science series. CRC Press, 2013.

15 T.K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Number 23 in Cambridge monographs on applied and computational mathematics. Cambridge University Press, 2007.

16 M.P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

17 M.P. do Carmo. *Riemannian Geometry*. Birkhäuser, 1992.

18 Paul Erdős. Some remarks on the measurability of certain sets. *Bulletin of the American Mathematical Society*, 51(10):728–731, 1945.

19 Paul Erdős. On the Hausdorff dimension of some sets in Euclidean space. *Bulletin of the American Mathematical Society*, 52(2):107–109, 1946. doi:10.1090/S0002-9904-1946-08514-6.

20 H. Federer. Curvature measures. *Transactions of the America Mathematical Society*, 93:418–491, 1959.

21 H. Federer. *Geometric Measure Theory*. Classics in Mathematics. Springer, 1996.

22 Charles Fefferman, Sergei Ivanov, Yaroslav Kurylev, Matti Lassas, and Hariharan Narayanan. Reconstruction and interpolation of manifolds. I: The geometric Whitney problem. *Foundations of Computational Mathematics*, 2019. doi:10.1007/s10208-019-09439-7.

23 Charles Fefferman, Sergei Ivanov, Matti Lassas, and Hariharan Narayanan. Fitting a manifold of large reach to noisy data. *Journal of Topology and Analysis*, 17(02):315–396, July 2023. doi:10.1142/s1793525323500012.

24 M.W. Hirsch. *Differential Topology*. Springer-Verlag: New York, Heidelberg, Berlin, 1976.

25 Norbert Kleinjohann. Nächste Punkte in der Riemannschen Geometrie. *Mathematische Zeitschrift*, 176(3):327–344, 1981. doi:10.1007/BF01214610.

26 André Lieutier and Mathijs Wintraecken. Manifolds of positive reach, differentiability, tangent variation, and attaining the reach. In Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri, editors, *42nd International Symposium on Computational Geometry (SoCG 2026)*, volume 367 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 74:1–74:16, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2026.74.

27 Alexander Lytchak. On the geometry of subsets of positive reach. *manuscripta mathematica*, 115(2):199–205, 2004. doi:10.1007/s00229-004-0491-8.

28 Alexander Lytchak. Almost convex subsets. *Geometriae Dedicata*, 115(1):201–218, 2005. doi:10.1007/s10711-005-5994-2.

29 Marina Meilă and Hanyu Zhang. Manifold learning: What, how, and why. *Annual Review of Statistics and Its Application*, 11(1):393–417, April 2024. doi:10.1146/annurev-statistics-040522-115238.

30 David Milman and Zeev Waksman. On topological properties of the central set of a bounded domain in $\mathbb{R}^m$. *Journal of Geometry*, 15(1):1–7, 1980. doi:10.1007/BF01919351.

31 Sumner Byron Myers. Connections between differential geometry and topology I Simply connected surfaces. *Duke Math. J.*, 1(1):376–391, 1935.

32 Sumner Byron Myers. Connections between differential geometry and topology II Closed surfaces. *Duke Math. J.*, 2(1):95–102, 1936.

33 P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 39(1-3):419–441, 2008. doi:10.1007/s00454-008-9053-2.

34 Henri Poincaré. Sur les lignes géodésiques des surfaces convexes. *Transactions of the American Mathematical Society*, 6(3):237–274, 1905.

35 Ludovic Rifford. From the poincaré "lignes de partage" to the convex earth theorem. https://math.univ-cotedazur.fr/~rifford/Papiers_en_ligne/IHP100_Talk_LR.pdf.

36 Barak Sober and David Levin. Manifold approximation by moving least-squares projection (MMLS). *Constructive Approximation*, pages 1–46, 2019. doi:10.1007/s00365-019-09489-8.

37 Yuan Wang and Bei Wang. Topological inference of manifolds with boundary. *Computational Geometry*, 88:101606, 2020. doi:10.1016/j.comgeo.2019.101606.

38 John Henry Constantine Whitehead. On the covering of a complete space by the geodesics through a point. *Annals of Mathematics*, 36(3):679–704, 1935.

On the Size of k-Irreducible Triangulations

Vincent Delecroix
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France

Oscar Fontaine
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France

Arnaud de Mesmay
LIGM Laboratoire Informatique Gaspard Monge Univ Gustave Eiffel, CNRS, LIGM, F-77454 Marne-la-Vallée, France

Abstract

A triangulation of a surface is k-irreducible if every non-contractible curve has length at least k and any edge contraction breaks this property. Equivalently, every edge belongs to a non-contractible curve of length k and there are no shorter non-contractible curves. We prove that a k-irreducible triangulation of an orientable surface of genus g has $O(k^2g)$ triangles, which is optimal. This is an improvement over the previous best bound $k^{O(k)}g^2$ of Gao, Richter and Seymour [*Journal of Combinatorial Theory, Series B, 1996*].

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces; Theory of computation → Computational geometry

Keywords and phrases surface, irreducible triangulation, system of curves, minimal position, systolic geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.38

Related Version *Full Version*: `https://arxiv.org/pdf/2603.20030` [21]

Funding This work was funded by the ANR-SNF project SUGAR (ANR-25-CE40-0416, SNF 200021E_238147).
Vincent Delecroix: Funded by ANR MOST (ANR-23-CE40-0020) and ANR CarteEtPlus (ANR-23-CE48-0018).

Acknowledgements We are grateful to anonymous reviewers for helpful remarks and suggestions.

1 Introduction

The main objects of study in this article are **(surface) triangulations**. There are many inequivalent definitions of those, and for us they will be simplicial complexes homeomorphic to surfaces. Rephrased in a graph-theoretical language, this means that a triangulation is a graph embedded in a surface where all the faces are topological disks and have degree three, and the graph is **simple**: multiple edges and loops are forbidden.

In 1922, Steinitz [41] proved that all triangulations of the 2-dimensional sphere could be obtained from the tetrahedron by *splitting vertices*. Nowadays, it is more common to consider the inverse operation of **contracting an edge**, which consists in identifying the endpoints of an edge and collapsing the two adjacent triangles. Such a contraction must preserve the properties of triangulations, and in particular must not induce multiple edges and loops. Thus, Steinitz proved that the tetrahedron is the unique **irreducible triangulation** of the sphere: every triangulation of the sphere can be reduced to the tetrahedron by iteratively contracting edges, and the tetrahedron cannot be contracted further.

Barnette initiated the study of the analogous problem on the projective plane [3], which led to the classification of irreducible triangulations of surfaces up to Euler genus 4 by Sulanke [42] by means of the computer program `surftri`. It is now known that for every

© Vincent Delecroix, Oscar Fontaine, and Arnaud de Mesmay;
licensed under Creative Commons License CC-BY 4.0

surface of Euler genus g, the number of irreducible triangulations is finite and that they have at most $13g - 4$ vertices, see Joret and Wood [29]. The latter improves previous bounds by Barnette and Edelson [2] and by Nakamoto and Ota [38]. This bound has been extended to surfaces with boundaries by Boulch, Colin de Verdière and Nakamoto [6].

A closed curve on a surface is **non-contractible** if it is not homotopic (intuitively, if it cannot be deformed continuously) to a point. Any closed walk on the edges of a triangulation determines a closed curve on the underlying surface. The **edge-width** of a triangulation is the length of the shortest (by the number of edges) non-contractible closed walk. A triangulation is **k-irreducible** if it has edge-width at least k and it is irreducible for this property: any edge contraction reduces this edge-width. Equivalently, every edge belongs to a non-contractible curve of length k and there are no shorter non-contractible curves. We refer to the PhD thesis of Melzer for a proof of this equivalence [36, Lemma 3.1] and for a thorough introduction to k-irreducible triangulations, in particular the different motivations for which they have been introduced.

Since loops and multiple edges are forbidden, triangulations of surfaces of positive Euler genus have edge-width at least 3. From the definitions, it is easy to see that irreducible triangulations and 3-irreducible triangulations match. Therefore, k-irreducible triangulations generalize irreducible triangulations, and one is naturally led to wonder about their combinatorial properties.

This problem was first studied by Malnič and Nedela [34] who proved that the number of k-irreducible triangulations of a surface is finite (they also observed that this follows from a variant of Robertson-Seymour theory), see also Juvan, Malnič and Mohar [30]. The current best known bound, due to Gao, Richter and Seymour [25] is that a k-irreducible triangulation of a surface of Euler genus g has $k^{O(k)}g^2$ edges. In that paper, the authors say that they expect the correct bound to depend linearly on g instead of quadratically. In his aforementioned PhD thesis, Melzer [36, p. 18] states "Our expectation would be something like $O(k^2g)$, but we lack any evidence at all to conjecture". Our main result confirms the expectations of Gao, Richter, Seymour and Melzer for orientable surfaces.

▶ **Theorem 1.** *Any k-irreducible triangulation of an orientable surface S of genus g has at most $966k^2g = O(k^2g)$ edges.*

The bound $O(k^2g)$ in Theorem 1 is tight up to the value of the constant which we have not tried to optimize: intuitively it can be achieved by taking g connected sums of tori resembling k by k grids. We refer to Melzer [36, Section 6.2] for precise descriptions of families of k-irreducible triangulations of size $\Theta(k^2g)$. One interesting follow-up question would be to determine the correct constant in some asymptotic regimes in g and k.

Note that an immediate application of the Euler formula yields similar bounds for the number of vertices and faces.

Irreducible systems of curves. Our proof of Theorem 1 is actually a consequence of a stronger result on system of curves that we introduce now. We investigate k-irreducible triangulations from the powerful perspective initiated by (de Graaf and) Schrijver in a series of papers (see e.g., [39, 40, 19, 20]), which consists in encoding the metric structure of an embedded graph using a system of curves. Note that this approach was already implicit in Steinitz's proof of his aforementioned theorem, as explained in Chang [10].

For the case of k-irreducible triangulations, this is done as follows. One first observes that instead of considering shortest non-contractible closed walks, one could consider instead shortest non-contractible closed curves crossing the triangulation exactly at the vertices.

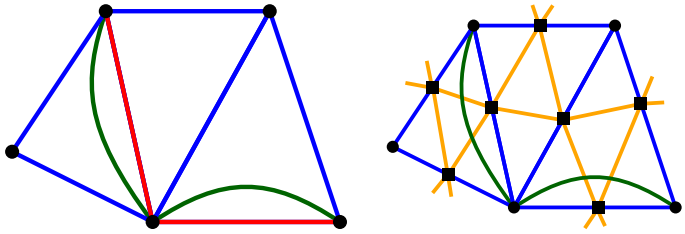


Figure 1 The triangulation in blue, a walk in red, its corresponding noose in green and the medial graph in orange.

Such curves are commonly called **nooses** in structural graph theory. If one defines the length of those curves as this number of intersections, one sees that any non-contractible closed walk in a triangulation can be deformed to a non-contractible noose of the same length and vice-versa, see Figure 1, left. The length of a shortest non-contractible noose is called the **face-width** (also called **representativity** in, e.g. [25]), and therefore we can equivalently define k-irreducibility as having face-width at least k and being irreducible for this property.

The **medial graph** M of a triangulation T of a surface S is the embedded graph in S which has a vertex in the middle of each edge of T and in each face f of T we add edges between vertices of M that belong to consecutive edges of f. See Figure 1, right for an illustration. Note that M is a simple face-bipartite 4-regular graph and that the length of a noose c for T is equal to half the number of transverse intersections it has with M. Because M is 4-regular, it can be considered as a **system of curves** $\mathcal{C} = (c_1, \dots, c_n)$. The **crossing number** of a system of curves $\mathcal{C}$, denoted by $\mathrm{cr}(\mathcal{C})$ is its number of intersections. The **length** of a curve c with respect to $\mathcal{C}$ is the number of intersections of c and $\mathcal{C}$, denoted by $\mathrm{cr}(c, \mathcal{C})$.

The strength of this perspective is that this allows to understand metric deformations of T (e.g., edge contractions) under the lens of topological deformations (homotopies) of $\mathcal{C}$. For instance, it follows from the work of Schrijver [40] that on an orientable surface, the system of curves corresponding to a minor-minimal graph of face-width k is necessarily in **minimal position**: it contains no contractible curves and its crossing number $\mathrm{cr}(\mathcal{C})$ is minimized over all families $\mathcal{C}'$ homotopic to $\mathcal{C}$ (we refer to Section 2 for the background on homotopies and a precise statement of Theorem 4 by Schrijver).

A **smoothing** of a system of curves $\mathcal{C}$ is the system of curves obtained from $\mathcal{C}$ by removing a vertex v and reconnecting the incident strands in one of two ways as in Figure 2. Note that a smoothing may change the number of components.

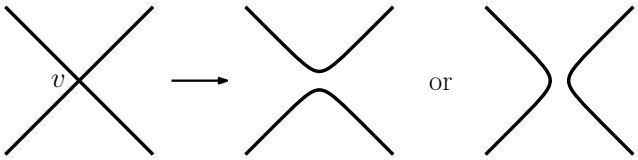


Figure 2 A smoothing at v.

A system of curves $\mathcal{C}$ on a surface S is a **k-irreducible** system of curves for some $k \geqslant 1$ if it is connected and:

- Every non-contractible closed curve c on S transverse to $\mathcal{C}$ satisfies $\mathrm{cr}(c, \mathcal{C}) \geqslant k$, and
- for every system of curves $\mathcal{C}'$ obtained by a smoothing of $\mathcal{C}$, there is a non-contractible closed curve c such that $\mathrm{cr}(c, \mathcal{C}') < k$.

Finally, a system of curves on a surface is **face-bipartite** if there is a 2-coloration of its faces (connected components of the complement) such that any two adjacent faces have different colors. We now have all the ingredients to state our second theorem.

▶ **Theorem 2.** *Let $\mathcal{C}$ be a face-bipartite k-irreducible system of curves on an orientable surface of genus g. The crossing number of $\mathcal{C}$ is at most $120k^2g = O(k^2g)$.*

Connections to systolic geometry. Theorems 1 and 2 provide discrete counterparts to results in systolic geometry. This subfield of differential geometry investigates topological spaces endowed with continuous metrics (typically Riemannian) and aims at understanding the structure and properties of their shortest non-contractible curves, called **systoles**. We refer to the book of Katz [31] for an introduction.

The famed systolic inequality of Gromov [26] states that for any surface of genus g and area A, the systole has length $O(\sqrt{A/g}\log g)$. This is known to be tight: Brooks [8] and Buser and Sarnak [9] provided families of hyperbolic surfaces achieving this bound. There are discrete analogues of these results for embedded graphs [28, 16, 33], showing that the edge-width of a triangulation with n triangles is similarly bounded by $O(\sqrt{n/g}\log g)$ and that this is asymptotically tight. Equivalently, this shows that any triangulation of edge-width k has $\Omega(k^2g/\log^2 g)$ triangles[1]. Bounding the size of k-irreducible triangulations then amounts to looking for a *reverse systolic inequality*, bounding how far from this lower bound a locally extremal object for the systolic inequality can be. From this perspective, it is noteworthy that the logarithmic factor, which is known to be required for the systolic inequality, disappears in our bounds in Theorems 1 and 2, even though our bounds are also tight.

We are not aware of any works investigating continuous versions of such a reverse systolic inequality. One difficulty is that the space of Riemannian metrics on a surface is not compact, and extremal surfaces for the systolic inequality in negative Euler characteristic are expected to exhibit singularities, see for example Gromov [26, pp.62-65]. This can be addressed by adding additional assumptions on the curvature, restricting to hyperbolic surfaces as in Fortier Bourque and Rafi [7], or nonpositively curved surfaces as in Katz and Sabourau [32]. Since the discrete setting of triangulations does not suffer from these analytic issues, it provides a convenient testbed for the study of locally extremal surfaces. The framework of geodesic currents of Bonahon [5] could provide an interesting middle-ground between these two worlds, keeping a continuous flavour but with strong compactness properties.

Related works. Many articles in computational geometry in the last twenty years have been targeted at understanding topologically meaningful curves on surfaces, from the perspectives of both algorithms and combinatorics. We refer to the dedicated chapter by Éric Colin de Verdière in the handbook of computational geometry [15] for a survey of this active field of research. Our proof techniques deal specifically with families of curves in minimal position on surfaces, a problem for which polynomial-time algorithms have been recently provided by Chang and de Mesmay [12], and Dubois [23]. The study of systolic properties of surfaces through the lens of such systems of curves has been instrumental in recent works of Cossarini [17] and Cossarini and Sabourau [18]. The medial graph construction connects homotopy of curves with electrical moves, as studied in Chang and Erickson [13], Chang, Cossarini and Erickson [11] and Aranguri, Chang and Fridman [1]. Very recently, Delecroix, Fontaine and Lazarus [22] have made algorithmic Schrijver's concept of *kernel* [40], which generalizes our k-irreducible triangulations to metric structures which are extremal with respect to the entire set of lengths of non-contractible curves (the *length spectrum*).

[1] Note that the growth is sensibly different if one considers the number of vertices instead of the number of triangles, see Melzer [36, Section 2.2]

Organization of the paper. After some preliminaries in Section 2, we first explain how Theorem 2 implies Theorem 1 in Section 3. Then the main ingredient is Proposition 7 proven in Section 4. Namely given a k-irreducible system of curves, it shows the existence of $O(g)$ curves of length at most $2k$ filling S. This immediately yields a first bound (Theorem 13), albeit one not linear in g. Section 5 introduces additional combinatorial tools to obtain the tight bound of Theorem 2. Due to line restrictions, some proofs are deferred to the full version [21], which provides slightly more general results and a treatment of one case of non-orientable surface.

2 Preliminaries

In this section we provide the main definitions used throughout the paper. The central objects of study are topological surfaces and curves on them whose definitions are recalled in Section 2.1. A (discrete) geometry on a surface allowing to measure lengths can then be defined in two related ways: either by the mean of an embedded graph also called combinatorial surface (see Section 2.2), or using a system of curves as in Section 2.3.

2.1 Surfaces and curves

In this paper, we use the term **surface** for a closed orientable topological surface. It is always denoted by S and is sometimes implicit. We refer to Massey [35] for topological background.

Curves. A **curve** on a surface S is a continuous map $c : [0, 1] \to S$ and a **closed curve** on S is a continuous map $c : \mathbb{S}^1 \to S$, where $\mathbb{S}^1$ is the one-dimensional circle $\mathbb{R}/\mathbb{Z}$. Since all the curves in this article are closed, we will often just write curve instead of closed curve. Two closed curves c and c' on S are **(freely) homotopic** if there is a continuous map $h : [0, 1] \times \mathbb{S}^1 \to S$ such that $h(0, t) = c(t)$ and $h(1, t) = c'(t)$. Being freely homotopic is an equivalence relation on closed curves on S and we refer to **(free) homotopy classes** for an equivalence class of closed curves with respect to free homotopy. We always denote curves with latin letters and homotopy classes with greek letters. A closed curve is **contractible** if it is homotopic to a constant curve. A curve is **simple** if it does not self-intersect. A closed curve is **primitive** if it is not homotopic to the k-fold concatenation of a curve c for some $k \geqslant 2$. In particular, a primitive curve is not contractible.

2.2 Combinatorial surfaces

Combinatorial surfaces. A **combinatorial surface** is an embedding $\phi : G \to S$ of a graph $G = (V, E)$, possibly with multiple edges and loops, in a surface S, such that $S \setminus \phi(G)$ is a disjoint union of open disks. These disks are called the **faces** of G. The counterclockwise sequence of arcs (i.e., oriented edges) in the boundary of a face is called a **face boundary** or **facial walk**. The **degree of a face** is the length of its facial walk.

Two combinatorial surfaces $\phi : G \to S$ and $\phi' : G' \to S'$ are **isomorphic** if there exists a graph isomorphism $\theta : G \to G'$ and a homeomorphism $\psi : S \to S'$ such that $\phi' \circ \theta = \psi \circ \phi$. The isomorphism class of a combinatorial surface can be conveniently encoded by a **rotation system** describing the circular orderings of the (oriented) edges around each vertex. See Mohar and Thomassen [37] for more details on this structure. Most of the time, the embedding ϕ is implicit and we identify G and the corresponding combinatorial surface.

Let $\phi : G \to S$ be a combinatorial surface. The **length (with respect to G)** of a curve c in S that crosses only the vertices of G and not its edges is the number of times this curve passes through a vertex of G. It is denoted $\mathrm{cr}(c, G)$. The **systole** of G is the minimum length

over all non-contractible closed curves c on S passing only through vertices of G of $\mathrm{cr}(c, G)$. We will use the common abuse of language to also call systole such a non-contractible closed curve of minimal length when it leads to no confusion.

Triangulation. A combinatorial surface (S, G) is a **triangulation** if G is simple (no loops nor multiple edges) and every face has degree 3. A triangulation T on a surface S is **k-irreducible** if

1. the systole of T is equal to k
2. for every edge $e = (uv)$ of T, there is a non-contractible curve c of length k on T crossing consecutively u and v.

▶ Remark 3. In line with the introduction, we define our triangulations to be simplicial, but note that this is implied by k-irreducibility for $k \geqslant 3$. Indeed, if there is a loop or a bigon formed of two parallel edges, it has to be contractible and thus bounds a disk, but then the edges inside that disk cannot belong to a systole as there would be a shortcut, violating k-irreducibility.

2.3 Systems of curves

Systems of curves. A **system of curves** on a surface S is a family $\mathcal{C} = (c_1, \ldots, c_p)$ of non-contractible closed curves on S in *general position*, i.e., they only cross transversely and at most two curves cross at a point.

An **intersection** of $\mathcal{C}$ is a pair $(i, s) \neq (j, t)$ such that $c_i(s) = c_j(t)$. Because a system of curves is in general position, the set of intersections is an isolated subset of points in S. The **crossing number of $\mathcal{C}$** is its number of intersections. Note that $\mathcal{C}$ defines a 4-regular embedded graph on S whose vertex set is the intersections of $\mathcal{C}$ and its edges are segments of the curves c_i. Conversely, a 4-regular embedded graph $\phi : G \to S$ determines a system of curves by "going straight" at each crossing. We will freely use this identification and use the vocabulary of embedded graphs, i.e., vertices, edges and faces for a system of curves $\mathcal{C}$. Note however that in this context the embedded graph might not be a combinatorial surface because its complement could have topology. a system of curves is **connected** if the corresponding graph is connected.

Two systems of curves $\mathcal{C} = \{c_1, \ldots, c_p\}$ and $\mathcal{C}' = \{c'_1, \ldots, c'_p\}$ are **homotopic** if for all i in $[1, p]$, c_i is freely homotopic to c'_i. We say that a system of curves $\mathcal{C}$ is in **minimal position** if it contains no contractible curves and has the minimal crossing number over all systems of curves homotopic to $\mathcal{C}$. We say that two curves **intersect essentially** if they intersect at least once in minimal position.

Length of curves. Let $\mathcal{C}$ be a system of curves on S and c be a curve transverse to $\mathcal{C}$. The **length** of c with respect to $\mathcal{C}$ is the number of crossings of c with $\mathcal{C}$. We recall that this quantity is denoted $\mathrm{cr}(c, \mathcal{C})$. The **length** of a free homotopy class γ is the minimal number of crossings of a representative c of γ with $\mathcal{C}$. We use the notation $\mathrm{cr}(\gamma, C)$ as a shortcut for $\min_{c \in \gamma} \mathrm{cr}(c, C)$. For $\mathcal{C}$ in minimal position, we define the **systole** of $\mathcal{C}$ to be $\min_{\gamma} \mathrm{cr}(\gamma, \mathcal{C})$ where γ runs over all non-trivial free homotopy classes of closed curves in S. Here again, we sometimes also call a closed curve achieving this minimum length a systole.

We say that a system of curves $\mathcal{C}$ in minimal position on a surface S is **filling** if for any non-trivial homotopy class γ we have $\mathrm{cr}(\gamma, \mathcal{C}) > 0$. Equivalently, a system of curves $\mathcal{C}$ in minimal position on a surface S is filling if and only if it corresponds to a combinatorial surface (i.e. the complement of the curves is a disjoint union of topological disks).

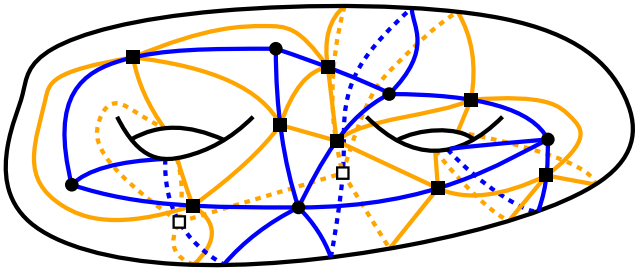


Figure 3 A graph G in blue on a genus 2 surface and its medial graph in orange.

***k*-irreducible systems of curves.** Recall from the definition that a system of curves is k-irreducible for some $k \geqslant 1$ if it is connected and:

- for every non-contractible closed curve c on S we have $\mathrm{cr}(c, \mathcal{C}) \geqslant k$, and
- for every system of curves $\mathcal{C}'$ obtained by a smoothing (pictured in Figure 2) of $\mathcal{C}$, there is a non-contractible closed curve c such that $\mathrm{cr}(c, \mathcal{C}') < k$.

A system of curves $\mathcal{C}$ is **tight** if it does not contain a contractible curve disjoint from the other curves and if for any smoothing $\mathcal{C}'$ of $\mathcal{C}$ there is a curve in minimal position with respect to $\mathcal{C}$ and $\mathcal{C}'$ which is shorter with respect to $\mathcal{C}'$ than with respect to $\mathcal{C}$. This is an equivalent reformulation of a notion introduced by Schrijver, who proved the following theorem:

▶ **Theorem 4** ([39, Theorem 5]). *A system of curves is tight if and only if it is made of primitive curves in minimal position.*

It is immediate that a k-irreducible system of curves is tight, and therefore it is in minimal position. Note also that a k-irreducible system of curves is necessarily filling.

▶ Remark 5. In this article, we often alternate between different metric models. In order to minimize confusion, we use the following guidelines:

- in a triangulation T, we consider curves which are walks on the edges of the triangulation, and their lengths are the number of edges that they use.
- in a combinatorial surface G, we consider curves which cross G only at vertices, and their lengths are the number of vertices that they cross.
- in a system of curves $\mathcal{C}$, we consider curves which are in general position with respect to $\mathcal{C}$, and their lengths are their number of intersections with $\mathcal{C}$.

3 From triangulations to system of curves

The goal of this section is to prove how Theorem 1 on k-irreducible triangulations follows from Theorem 2 on k-irreducible system of curves.

Let T be a k-irreducible triangulation on S with vertex set V. We first explain how to associate a system of curves to T, and we then study its properties. Let G be a subgraph of T with the same vertex set V and minimal (with respect to edge inclusion) for the property that every non-contractible curve c intersecting G only at V satisfies $\mathrm{cr}(c, G) \geqslant k$. Let M be the medial graph of G: recall that it is the graph with a vertex at the midpoint of every edge of G and two midpoints are connected by an edge of M whenever their supporting edges in G are consecutive in a face of G, see Figure 3. The graph M is 4-regular and is thus naturally associated to a system of curves $\mathcal{C}$ on S. We call $\mathcal{C}$ a **system of curves associated to T** (note that there can be many of them, depending on the choice of G). As proven in [40], for any homotopy class γ of curves in S we have $\mathrm{cr}(\gamma, \mathcal{C}) = 2\,\mathrm{cr}(\gamma, G)$.

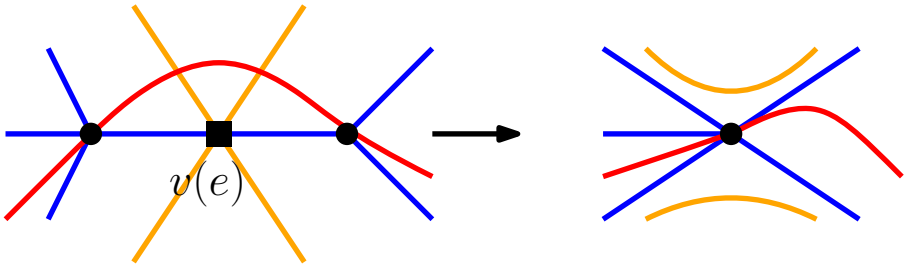


Figure 4 The contraction of an edge in G. The blue graph is G, the orange one is $\mathcal{C}$ and the red curve is c.

▶ **Proposition 6.** *Let T be a k-irreducible triangulation on S. Then any system of curves associated to T is face-bipartite, $2k$-irreducible and in minimal position.*

Proof. Let us denote as before by G an (edge inclusion-wise) minimal subgraph of T such that the length of a shortest non-contractible noose on G is still k, and by $\mathcal{C}$ the system of curves associated to G. As $\mathrm{cr}(c, G) \geqslant k$ for every non-contractible curve c, $\mathrm{cr}(\gamma, \mathcal{C}) = 2\,\mathrm{cr}(\gamma, G) \geqslant 2k$ for every non-trivial homotopy class γ.

Let $\mathcal{C}'$ be a smoothing of $\mathcal{C}$. As shown in [40], a smoothing of $\mathcal{C}$ either corresponds to an edge deletion or an edge contraction in G. We work separately on these two cases.

1. Assume that $\mathcal{C}'$ is obtained from $\mathcal{C}$ by a smoothing corresponding to deleting the edge e. Let $G' = G \setminus \{e\}$. Then by definition of G, there is a non-contractible curve c such that $\mathrm{cr}(c, G') < k$. Thus $\mathrm{cr}(\gamma, \mathcal{C}') \leqslant 2 \cdot \mathrm{cr}(c, G') < 2k$ where γ is the homotopy class of c.
2. Assume that $\mathcal{C}'$ is obtained from $\mathcal{C}$ by a smoothing corresponding to contracting the edge $e = (uv)$. As T is a k-irreducible triangulation, there is a systole c of T going through u and v consecutively. Thus this systole is also a systole of G with the same property and is also a systole of $\mathcal{C}$ crossing two consecutive edges of $\mathcal{C}$ around the vertex $v(e)$ (see Figure 4). Thus, the smoothing reduces the length of c and $\mathrm{cr}(\gamma, \mathcal{C}') < 2k$ where γ denotes the homotopy class of c.

Since G is connected, so is $\mathcal{C}$. Therefore the system of curves $\mathcal{C}$ is $2k$-irreducible. It follows from Theorem 4 that it is in minimal position. Finally, it is face-bipartite by construction: the faces of the medial graph are either faces or vertices of T. ◀

We are now ready to prove Theorem 1 assuming Theorem 2. We restate it for convenience.

▶ **Theorem 1.** *Any k-irreducible triangulation of an orientable surface S of genus g has at most $966k^2g = O(k^2g)$ edges.*

Proof. As before, let G be a minimal subgraph of T such that the length of a shortest non-contractible noose on G is k, and let $\mathcal{C}$ be its associated system of curves. By Proposition 6, any system of curves associated to T is a $2k$-irreducible system of curves. Thus by Theorem 2, the crossing number $\mathrm{cr}(\mathcal{C})$ of $\mathcal{C}$ is at most $120(2k)^2g = 480k^2g$.

Now let M be the medial graph of G from which is built the system of curves $\mathcal{C}$. It is such that $|V(M)| = \mathrm{cr}(\mathcal{C})$ and because it is quartic $|E(M)| = 2\,\mathrm{cr}((C))$. The graph M is face bipartite and the classes of the bipartition are in bijection with the vertices $V(G)$ and the faces $F(G)$ of G. These bijections preserve the degree and hence $\sum_{v \in V(G)} \deg_G(v) = |E(M)| = 2\,\mathrm{cr}(\mathcal{C})$. Now, by Proposition 6, $\mathcal{C}$ is in minimal position. In particular, $\mathcal{C}$ has

neither monogon nor bigon. In other words, each face of M has degree at least 3. Thus $\sum_{v \in V(G)} \deg_G(v) \geqslant 3|V(G)|$. We conclude that $|V(T)| = |V(G)| \leqslant \frac{2}{3}\mathrm{cr}(\mathcal{C}) \leqslant \frac{2}{3}480k^2g = 320k^2g$, and $|E(T)| \leqslant 3|V(T)| + 6g - 6 \leqslant 966k^2g$. ◀

4 Short filling systems for k-irreducible systems of curves

The main result of this section is the following proposition, which extracts from k-irreducibility a more topological property: the existence of a filling family of short curves.

▶ **Proposition 7.** *Let $\mathcal{C}$ be a face-bipartite k-irreducible system of curves. Then there is a set of curves of lengths at most $2k$ with respect to $\mathcal{C}$ which is in minimal position and filling.*

The proof of this proposition relies partially on the notion of universal cover. We refer to Massey [35, Chapter 5] for an introduction to this classical object in algebraic topology and to Farb and Margalit [24, Section 1.2] for the specifics in the case of curves on surfaces. Given a curve $c : \mathbb{S}^1 \to S$ in a surface S, a *lift* $\tilde{c} : \mathbb{R} \to \tilde{S}$ of c is a map to the universal cover $\tilde{S}$ that commutes with the projections $\mathbb{R} \to S^1$ and $\tilde{S} \to S$. When the lift $\tilde{c}$ is simple we call it a *line*.

Proof. We first observe that we can assume that k is even. Indeed, if this is not the case, since $\mathcal{C}$ is k-irreducible and face-bipartite, $\mathcal{C}$ is $(k-1)$-irreducible. We will build a set of curves of lengths at most $2k$ made of systoles and some curves that are obtained by concatenation of two systoles, and then show that these curves are filling.

The following claim is standard in geometry of curves on surfaces. As it is used many times throughout our proof, we provide a standalone proof.

▷ Claim 8. Let c and c' be non-contractible curves on S in minimal position. Then any two lifts $\tilde{c}$ and $\tilde{c'}$ in the universal cover $\tilde{S}$ are lines that intersect at most once.

Proof. Assume that the lift $\tilde{c}$ has a self-intersection. Then it forms a monogon: there is a disk in $\tilde{S}$ bounded by a subline of $\tilde{c}$. By homotopy, one can remove this monogon contradicting the fact that c was in minimal position on S. Therefore, $\tilde{c}$ and $\tilde{c'}$ are lines in $\tilde{S}$.

Let us assume for contradiction that two lifts $\tilde{c}$ and $\tilde{c'}$ intersect twice. Then, since $\tilde{S}$ is contractible, they form a bigon: there is a disk in $\tilde{S}$ bounded by one subline α of $\tilde{c'}$ and one subline β of $\tilde{c}$. Now, the homotopy moving α to β projects to a homotopy on the surface, and leads to a new pair of curves intersecting less. This contradicts the hypothesis that c and c' were in minimal position. ◁

The next claim is the first step in defining the systoles we will work with.

▷ Claim 9. Let v be a crossing of $\mathcal{C}$ and e, e' two edges adjacent to v that are consecutive for the counter-clockwise ordering of edges around v. Then there is a systole that passes consecutively through e and e'.

Proof. Let $\mathcal{C}'$ be the system of curves obtained by smoothing v and connecting e with e'. As $\mathcal{C}$ is a k-irreducible system, there is a non-contractible curve c in minimal position with respect to both $\mathcal{C}$ and $\mathcal{C}'$ and such that $\mathrm{cr}(c, \mathcal{C}') < k$. Since $\mathcal{C}$ and $\mathcal{C}'$ differ from each other only at v, the curve c goes from e to e' in a neighborhood of v. If we pick c with the property that $\mathrm{cr}(c, \mathcal{C}') < k$ and $\mathrm{cr}(c, \mathcal{C})$ minimal, then it passes only once consecutively through e and e' and $\mathrm{cr}(c, \mathcal{C}') = \mathrm{cr}(c, \mathcal{C}) - 2$.

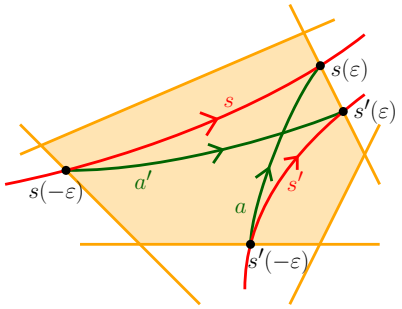


Figure 5 Wedding of two systoles s and s'.

As $\mathcal{C}$ is face-bipartite, all crossing numbers of curves with respect to $\mathcal{C}$ are even. Since k is even, we must have $\mathrm{cr}(c, \mathcal{C}') = k - 2$ and $\mathrm{cr}(c, \mathcal{C}) = k$. The curve c is therefore a systole, this concludes the proof of the claim. ◁

Let e_1, e_2, e_3 and e_4 be the four consecutive edges around a vertex v of $\mathcal{C}$. By Claim 9 there is a systole s_1 crossing consecutively e_1 and e_2. Similarly, there is a systole s_2 crossing consecutively e_2 and e_3. We denote by $\mathcal{S}$ a set of curves containing a pair of such systoles s_1 and s_2 for each vertex of $\mathcal{C}$. Note that by definition a systole is in minimal position with respect to $\mathcal{C}$.

We first handle the case where the surface is a torus.

▷ Claim 10. If the surface S is a torus and v is a vertex of $\mathcal{C}$ and s_1 and s_2 are the two corresponding systoles of $\mathcal{S}$, then s_1 and s_2 intersect essentially.

The proof is provided in the full version [21]. Since on a torus, any pair of curves intersecting essentially is filling, Claim 10 concludes the proof for the torus with the family $\mathcal{S}$. In the rest of the proof we assume that the surface has negative Euler characteristic. Therefore its universal cover is homeomorphic to an open disk and can be endowed with a boundary at infinity $\partial\tilde{S}$ which is a circle. The lifts of curves in $\tilde{S}$ are lines with endpoints on this boundary, and two curves are homotopic if and only if there exists lifts of the two curves having the same endpoints. Furthermore, no two lifts have exactly one endpoint in common. We refer to [24, Section 1].

We now define curves obtained by concatenations of two systoles. They will be used to fill the surface S. Let s and s' be two systoles in $\mathcal{S}$ that start in the same face f of $\mathcal{C}$ and such that $\mathcal{C} \cup \{s, s'\}$ is in minimal position. Up to homotopy we can further assume that the first crossings with the boundary of f occurs for both s and s' at time $\pm\varepsilon$ and that s and s' do not intersect in this initial face. Namely $s([-\varepsilon, \varepsilon]) \cap s'([-\varepsilon, \varepsilon]) = \emptyset$ and that the points $s(\varepsilon)$, $s(-\varepsilon)$, $s'(-\varepsilon)$ and $s'(\varepsilon)$ appear on the boundary of f in that counterclockwise order. Let a and a' be arcs in the face f that are in minimal position and go respectively from $s'(-\varepsilon)$ to $s(\varepsilon)$ and $s(-\varepsilon)$ to $s'(\varepsilon)$. These two arcs intersect exactly once. We define a new curve t by the concatenation $t := a \cdot s|_{[\varepsilon,-\varepsilon]} \cdot a' \cdot s'|_{[\varepsilon,-\varepsilon]}$, see Figure 5. We say that t is obtained by **wedding** s and s' if t is in minimal position with respect to itself. Equivalently, we can make a wedding if in the universal cover $\widetilde{S}$ of S the two lifts of s and s' starting from the same lift $\widetilde{f}$ of f do not intersect. By definition, a wedding has length $2k$ with respect to $\mathcal{C}$. If such a wedding is put in minimal position with respect to $\mathcal{C}$ by a homotopy, it might even get shorter.

We define the set $\mathcal{S}_2$ to be a set of representatives of the homotopy classes of all possible weddings of systoles in $\mathcal{S}$ such that $\mathcal{C} \cup \mathcal{S} \cup \mathcal{S}_2$ is in minimal position. The following claim proves that the curves in $\mathcal{S} \cup \mathcal{S}_2$ intersect every possible simple closed curve on the surface.

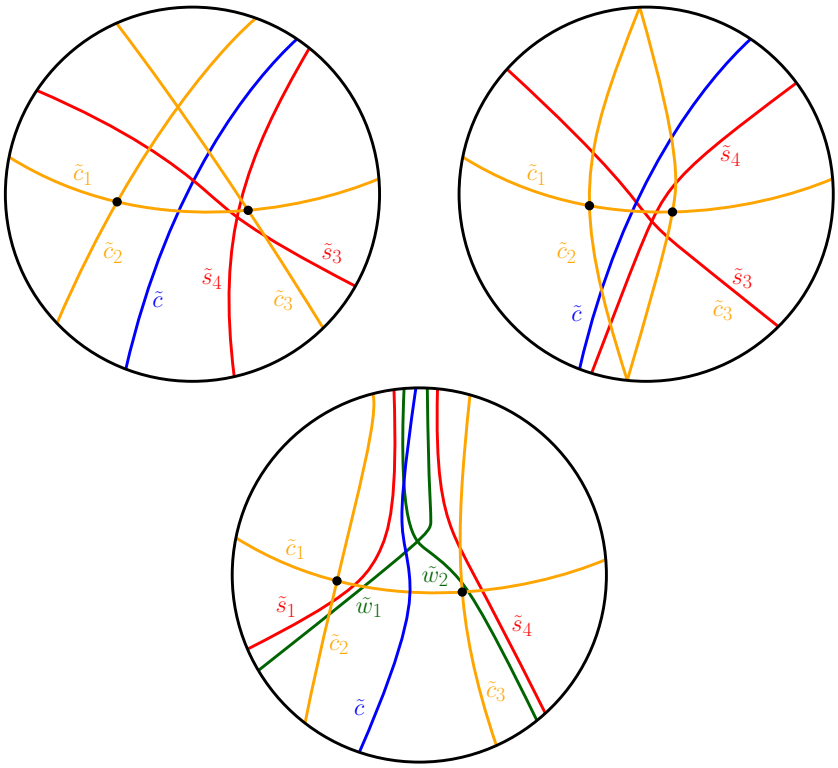


Figure 6 The three different cases leading to the proof of claim 11.

▷ Claim 11. Let c be a non-contractible simple closed curve. Then there exists $s \in (\mathcal{S} \cup \mathcal{S}_2)$ such that c and s intersect essentially.

Proof. We first put c in minimal position with respect to $\mathcal{C} \cup \mathcal{S} \cup \mathcal{S}_2$. Since $\mathcal{C}$ is filling, there is an edge $e = (uv)$ of $\mathcal{C}$ that intersects c. Let c_1 be the curve of $\mathcal{C}$ containing the edge e and c_2 and c_3 be the curve crossing c_1 at u and v respectively. We denote by s_1 and s_2 the two systoles in $\mathcal{S}$ corresponding to u and by s_3 and s_4 the ones corresponding to v. We will prove that c intersects either s_1, s_2, s_3, s_4 or one of their pairwise weddings.

In the universal cover, our local picture lifts as in Figure 6, yielding lines $\tilde{c_1}$, $\tilde{c_2}$, $\tilde{c_3}$, $\tilde{s_1}$, $\tilde{s_2}$, $\tilde{s_3}$, $\tilde{s_4}$ and $\tilde{c}$. The lines $\tilde{c_1}$,$\tilde{c_2}$, $\tilde{c_3}$ intersect pairwise at most once and are simple because $\mathcal{C}$ is in minimal position. We do a case by case analysis depending on the topology of $\tilde{c_2}$ and $\tilde{c_3}$. The only ingredient we use is the fact that each pair of these curves intersect at most once.

If $\tilde{c_2}$ and $\tilde{c_3}$ intersect, then it is immediate that $\tilde{c}$ intersects at least one of the curves $\tilde{s_1}$, $\tilde{s_2}$, $\tilde{s_3}$ or $\tilde{s_4}$, see top left Figure 6. The same happens if c_2 and c_3 are homotopic since in that case $\tilde{c_2}$ and $\tilde{c_3}$ have the same endpoints in $\partial\tilde{S}$. Since these intersections are forced by the endpoints on $\partial\tilde{S}$ and homotopies do not move these endpoints, these intersections on $\tilde{S}$ project to essential intersections on S. See top right Figure 6.

The last case is if $\tilde{c_2}$ and $\tilde{c_3}$ are disjoint (including at their endpoints) as in bottom Figure 6. Then the wedding w of s_1 and s_4 based at the added crossing lifts to two lines $\tilde{w_1}$ and $\tilde{w_2}$ which cross the lines $\tilde{c_1}$ and $\tilde{c_2}$, respectively $\tilde{c_2}$ and $\tilde{c_3}$ the same way that $\tilde{s_1}$ and $\tilde{s_4}$ do. In particular, this wedding exists since $\tilde{w_1}$ and $\tilde{w_2}$ cannot cross again due to $\tilde{c_1}, \tilde{c_2}$ and $\tilde{c_3}$ acting as barriers for $\tilde{s_1}$ and $\tilde{s_4}$. Since $\tilde{c}$ intersects $\tilde{e}$, it must intersect either $\tilde{w_1}$ or $\tilde{w_2}$ and thus c intersects w essentially. This concludes the proof. ◁

Therefore, the family $\mathcal{S} \cup \mathcal{S}_2$, intersects every possible simple non-contractible curve in the surface: it is filling. This claim concludes the proof. ◀

We encapsulate this last result in the following definition: we say that a system of curves $\mathcal{C}$ is **k-saturating** if there exists another system of curves $\mathcal{S}$ such that:

- $\mathcal{C} \cup \mathcal{S}$ is in minimal position,
- Each curve in $\mathcal{S}$ has length at most k, and
- The system of curves $\mathcal{S}$ is filling.

Therefore, Proposition 7 shows that a face-bipartite k-irreducible system of curves is $2k$-saturating. The following lemma shows that one can extract a filling family of size $O(g)$ from any filling family. This is certainly folklore, see for example Chen [14, Theorem 3.15]. We include a proof in the full version [21] for completeness.

▶ **Lemma 12.** *Let $\mathcal{C}$ be a filling system of curves in minimal position on a closed orientable surface S of genus $g \geqslant 1$. Then there is a sub-system $\mathcal{C}' \subset \mathcal{C}$ of at most $3g-1$ curves which is also filling.*

We can now combine our tools to obtain a first polynomial upper bound.

▶ **Theorem 13.** *Let $\mathcal{C}$ be a k-saturating system of curves. Then the crossing number of $\mathcal{C}$ is at most $2((3g-1)\cdot k)^2$.*

Proof. By the definition of a k-saturating system of curves and thanks to Lemma 12, there exists a filling family $\mathcal{S} = (s_i)_{i\in I}$ of closed curves of length at most k with respect to $\mathcal{C}$ and of size at most $3g-1$. In particular, for every $i \in I$, $\mathrm{cr}(s_i, \mathcal{C}) \leqslant k$.

Recall that by the definition of k-saturating, the system $\mathcal{C} \cup \mathcal{S}$ is in minimal position. We think of $\mathcal{S}$ as a 4-valent graph that is a combinatorial surface. In other words:

- Every curve of $\mathcal{C}$ intersect transversely the curves s_i, and
- Every intersection vertex of $\mathcal{C}$ is inside a face of $\mathcal{S}$.

We denote by F the faces of $\mathcal{S}$. For such a face f, let $p(f)$ denote the perimeter of f i.e. the number of crossings of $\mathcal{C}$ with the boundary of f. Then, by double counting, $\sum_{f\in F} p(f) = 2\,\mathrm{cr}(\mathcal{S}, \mathcal{C}) \leqslant 2 \cdot k \cdot \#\mathcal{S} \leqslant 2(3g-1)k$.

Similarly let $n(f)$ denote the number of vertices of $\mathcal{C}$ in the face f. Then $\mathrm{cr}(\mathcal{C}) = \sum_{f\in F} n(f)$. As $\mathcal{C}$ is in minimal position, the restriction of $\mathcal{C}$ to any face f is also in minimal position. This implies that any two curves in $\mathcal{C}$ cross at most once in f, since otherwise they would form a bigon and thus would not be in minimal position. Thus $n(f) \leqslant \dfrac{p(f)\cdot(p(f)-1)}{2}$. So

$$\mathrm{cr}(\mathcal{C}) = \sum_{f\in F} n(f) \leqslant \sum_{f\in F} \frac{p(f)^2}{2} - \frac{p(f)}{2} \leqslant \frac{1}{2}\left(\sum_{f\in F} p(f)\right)^2 \leqslant \frac{(2(3g-1)k)^2}{2}. \qquad \blacktriangleleft$$

Combining Proposition 7 and Theorem 13 provides us with a first bound of $O(k^2g^2)$ on the number of vertices in a k-irreducible system of curves, and thus on the number of edges in a k-irreducible triangulation via the reduction in Section 3. In the next section we add another ingredient to strengthen this bound to $O(k^2g)$.

▶ Remark 14. Note that the family $\mathcal{C}$ in Theorem 13 is not assumed to be k-irreducible, and thus this theorem could be of independent interest. It is easy to see that Theorem 13 is tight up to a constant factor, by considering a genus g orientable surface obtained by identifying opposite edges in a $2g$-gon, and a system of curves made of k parallel curves between each pair of opposite edges.

5 Upper bounds for k-irreducible systems of curves

In this section, we show:

▶ **Theorem 2.** *Let $\mathcal{C}$ be a face-bipartite k-irreducible system of curves on an orientable surface of genus g. The crossing number of $\mathcal{C}$ is at most $120k^2g = O(k^2g)$.*

To prove this result, we first establish a preliminary lemma on configuration of arcs in a disk. An **arc** in a disk is a curve with endpoints on the boundary of the disk. For a system of arcs $\mathcal{A}$ in minimal position in a disk, we denote by d the largest distance between a face of $\mathcal{A}$ and the boundary of the disk (where we measure lengths in terms of crossings with $\mathcal{A}$) and by n the number of intersections between pairs of arcs of $\mathcal{A}$. The following inequality allows us to bound n in terms of d and p.

▶ **Lemma 15.** *Let $\mathcal{A}$, p, d, n be as above. Then $(6d+7)p \geqslant n$.*

Our proof relies on a recent structural result of Hickingbotham, Illingworth, Mohar and Wood [27] and the fact that a graph of treewidth k with n vertices has less than kn edges (see, e.g., Baste, Noy and Sau [4]).

Proof. Let G be the intersection graph of $\mathcal{A}$: it has one vertex per arc of $\mathcal{A}$ and an edge between vertices corresponding to intersecting pair of arcs. By definition, G has p vertices and n edges.

Let M_G be the map graph of $\mathcal{A}$ which has one vertex for each face of $\mathcal{A}$, including the outer face, and edges between vertices corresponding to faces that share at least one vertex. Let $\mathrm{rad}(M_G)$ be its radius, that is, the minimum radius of a ball that covers M_G. Then, by [27, Theorem 5], we have $\mathrm{tw}(G) \leqslant 6 \cdot \mathrm{rad}(M_G) + 7$, where $\mathrm{tw}(G)$ denotes the tree-width of G.

Let v_∞ be the vertex of M_G corresponding to the external face. Then v_∞ is at distance at most d from every face of $\mathcal{A}$ in the dual graph of $\mathcal{A}$. It follows from the definitions that the distances in the map graph are smaller than the one in the dual graph, so v_∞ is at distance at most d in M_G from every point of M_G. Thus $\mathrm{rad}(M_G) \leqslant d$ and $\mathrm{tw}(G) \leqslant 6d+7$.

Since a graph of tree-width t and v vertices has strictly less than tv edges, $(6d+7)p \geqslant n$. ◀

We can now conclude the proof of Theorem 2.

Proof of Theorem 2. The proof starts like the proof of Theorem 13. By Proposition 7 and Lemma 12, a k-irreducible system of curves is $2k$-saturating and thus there is a filling family $\mathcal{S}$ of at most $3g-1$ closed curves of length at most $2k$ with respect to $\mathcal{C}$. We think of $\mathcal{S}$ as a 4-valent combinatorial surface with a family of faces F and denote by $n(f)$ the number of vertices of $\mathcal{C}$ in a face f, and by $p(f)$ the perimeter of f, i.e., the number of intersections of $\mathcal{C}$ and the boundary of f.

▷ Claim 16. For any face f of F, $n(f) \leqslant (3k+7) \cdot p(f)$

Proof. Assume that there is a face f such that $n(f) > (3k+7)p(f)$. By Lemma 15, $(6d+7)p(f) > (3k+7)p(f)$ and thus $d > k/2$. In this case, there is a point R in f so that any path from R to the boundary of f crosses more than $k/2$ curves of $\mathcal{C}$. This point R lies in a face of $\mathcal{C}$, and this implies that smoothing one of the vertices on the boundary of that face cannot change the length of any non-contractible closed curve of length k, since any such curve going through (two consecutive edges adjacent to) that vertex (as in Claim 9) must be longer than k. See Figure 7. Thus $\mathcal{C}$ is not a k-irreducible system of curves. ◁

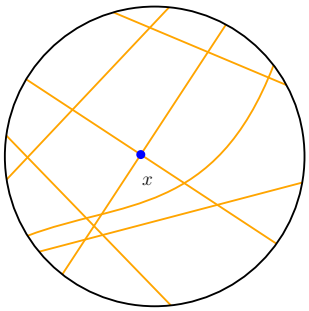


Figure 7 A system of arcs in a disk with a face x at distance 2 and a blue vertex that may be smoothed without changing the length of any curve of length smaller than 2.

We conclude using $\sum_{f\in F} n(f) = \mathrm{cr}(\mathcal{C})$ and $\sum_{f\in F} p(f) = 2\,\mathrm{cr}(\mathcal{S},\mathcal{C}) \leqslant 2(3g-1)2k$, which gives

$$\mathrm{cr}(\mathcal{C}) = \sum_{f\in F} n(f) \leqslant \sum_{f\in F} (3k+7)\cdot p(f) \leqslant (3k+7)2(3g-1)2k \leqslant 120k^2 g. \qquad \blacktriangleleft$$

References

1 Santiago Aranguri, Hsien-Chih Chang, and Dylan Fridman. Untangling planar graphs and curves by staying positive. In *Proceedings of the 33rd annual ACM-SIAM symposium on discrete algorithms, SODA 2022, Alexandria, VA, USA, both virtually and physically, January 9–12, 2022*, pages 211–225. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM); New York, NY: Association for Computing Machinery (ACM), 2021. doi:10.1137/1.9781611977073.11.

2 D. W. Barnette and Allan L. Edelson. All 2-manifolds have finitely many minimal triangulations. *Isr. J. Math.*, 67(1):123–128, 1989. doi:10.1007/BF02764905.

3 David Barnette. Generating the triangulations of the projective plane. *J. Comb. Theory, Ser. B*, 33:222–230, 1982. doi:10.1016/0095-8956(82)90041-7.

4 Julien Baste, Marc Noy, and Ignasi Sau. On the number of labeled graphs of bounded treewidth. *Eur. J. Comb.*, 71:12–21, 2018. doi:10.1016/j.ejc.2018.02.030.

5 Francis Bonahon. The geometry of Teichmüller space via geodesic currents. *Invent. Math.*, 92(1):139–162, 1988. doi:10.1007/BF01393996.

6 Alexandre Boulch, Éric Colin de Verdière, and Atsuhiro Nakamoto. Irreducible triangulations of surfaces with boundary. *Graphs Comb.*, 29(6):1675–1688, 2013. doi:10.1007/s00373-012-1244-1.

7 Maxime Fortier Bourque and Kasra Rafi. Local maxima of the systole function. *J. Eur. Math. Soc. (JEMS)*, 24(2):623–668, 2022. doi:10.4171/JEMS/1113.

8 Robert Brooks. Injectivity radius and low eigenvalues of hyperbolic manifolds. *J. Reine Angew. Math.*, 390:117–129, 1988. doi:10.1515/crll.1988.390.117.

9 P. Buser and P. Sarnak. On the period matrix of a Riemann surface of large genus (with an appendix by J. H. Conway and N. J. A. Sloane). *Invent. Math.*, 117(1):27–56, 1994. doi:10.1007/BF01232233.

10 Hsien-Chih Chang. *Tightening curves and graphs on surfaces*. PhD thesis, University of Illinois at Urbana-Champaign, 2018.

11 Hsien-Chih Chang, Marcos Cossarini, and Jeff Erickson. Lower bounds for electrical reduction on surfaces. In *35th international symposium on computational geometry, SoCG 2019, Portland, Oregon, USA, June 18–21, 2019. Proceedings*, page 16. Wadern: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. Id/No 25. doi:10.4230/LIPIcs.SoCG.2019.25.

12 Hsien-Chih Chang and Arnaud de Mesmay. Tightening curves on surfaces monotonically with applications. *ACM Trans. Algorithms*, 18(4):32, 2022. Id/No 36. doi:10.1145/3558097.

13 Hsien-Chih Chang and Jeff Erickson. Untangling planar curves. *Discrete Comput. Geom.*, 58(4):889–920, 2017. doi:10.1007/s00454-017-9907-6.

14 Changjie Chen. Index gap of the systole function, 2025. arXiv:2309.05801.

15 Éric Colin de Verdière. Computational topology of graphs on surfaces. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Toth, editors, *Handbook of Discrete and Computational Geometry*, chapter 23, pages 605–636. CRC Press LLC, third edition, 2018.

16 Éric Colin de Verdière, Alfredo Hubard, and Arnaud de Mesmay. Discrete systolic inequalities and decompositions of triangulated surfaces. *Discrete Comput. Geom.*, 53(3):587–620, 2015. doi:10.1007/s00454-015-9679-9.

17 Marcos Cossarini. Discrete surfaces with length and area and minimal fillings of the circle, 2020. arXiv:2009.02415.

18 Marcos Cossarini and Stéphane Sabourau. Minimal area of Finsler disks with minimizing geodesics. *J. Eur. Math. Soc. (JEMS)*, 26(3):985–1029, 2024. doi:10.4171/JEMS/1339.

19 Maurits de Graaf and Alexander Schrijver. Characterizing homotopy of systems of curves on a compact surface by crossing numbers. *Linear Algebra Appl.*, 226-228:519–528, 1995. doi:10.1016/0024-3795(95)00161-J.

20 Maurits de Graaf and Alexander Schrijver. Decomposition of graphs on surfaces. *J. Comb. Theory, Ser. B*, 70(1):157–165, 1997. doi:10.1006/jctb.1997.1747.

21 Vincent Delecroix, Oscar Fontaine, and Arnaud de Mesmay. On the size of k-irreducible triangulations, 2026. arXiv:2603.20030.

22 Vincent Delecroix, Oscar Fontaine, and Francis Lazarus. On the computation of schrijver's kernels, 2025. To appear in the Proceedings of SODA 2026. doi:10.48550/arXiv.2510.18597.

23 Loïc Dubois. Making multicurves cross minimally on surfaces. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, Royal Holloway, London, United Kingdom, September 2-4, 2024*, volume 308 of *LIPIcs*, pages 50:1–50:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.50.

24 Benson Farb and Dan Margalit. *A primer on mapping class groups*, volume 49 of *Princeton Math. Ser.* Princeton, NJ: Princeton University Press, 2011.

25 Z. Gao, R. B. Richter, and P. D. Seymour. Irreducible triangulations of surfaces. *J. Comb. Theory, Ser. B*, 68(2):206–217, 1996. doi:10.1006/jctb.1996.0064.

26 Mikhael Gromov. Filling Riemannian manifolds. *J. Differ. Geom.*, 18:1–147, 1983. doi:10.4310/jdg/1214509283.

27 Robert Hickingbotham, Freddie Illingworth, Bojan Mohar, and David R. Wood. Treewidth, circle graphs, and circular drawings. *SIAM J. Discret. Math.*, 38(1):965–987, 2024. doi:10.1137/22M1542854.

28 Joan P. Hutchinson. On short noncontractible cycles in embedded graphs. *SIAM J. Discrete Math.*, 1(2):185–192, 1988. doi:10.1137/0401020.

29 Gwenaël Joret and David R. Wood. Irreducible triangulations are small. *J. Comb. Theory, Ser. B*, 100(5):446–455, 2010. doi:10.1016/j.jctb.2010.01.004.

30 M. Juvan, A. Malnič, and B. Mohar. Systems of curves on surfaces. *J. Comb. Theory, Ser. B*, 68(1):7–22, 1996. doi:10.1006/jctb.1996.0053.

31 Mikhail G. Katz. *Systolic geometry and topology. With an appendix by Jake P. Solomon*, volume 137 of *Math. Surv. Monogr.* Providence, RI: American Mathematical Society (AMS), 2007.

32 Mikhail G. Katz and Stéphane Sabourau. Systolically extremal nonpositively curved surfaces are flat with finitely many singularities. *J. Topol. Anal.*, 13(2):319–347, 2021. doi:10.1142/S1793525320500144.

33 Ryan Kowalick, Jean-François Lafont, and Barry Minemyer. Filling triangulated surfaces. *Geom. Dedicata*, 202:373–386, 2019. doi:10.1007/s10711-018-00419-9.

34 A. Malnič and R. Nedela. k-minimal triangulations of surfaces. *Acta Math. Univ. Comen., New Ser.*, 64(1):57–76, 1995. URL: https://eudml.org/doc/119916.

35 William S. Massey. *A basic course in algebraic topology*, volume 127 of *Grad. Texts Math.* New York etc.: Springer-Verlag, 1991.

36 Sebastian Melzer. *k-irreducible triangulations of 2-manifolds*. PhD thesis, Technische Universität Dresden, 2019. URL: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-356439.

37 Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Baltimore, MD: Johns Hopkins University Press, 2001.

38 Atsuhiro Nakamoto and Katsuhiro Ota. Note on irreducible triangulations of surfaces. *J. Graph Theory*, 20(2):227–233, 1995. doi:10.1002/jgt.3190200211.

39 A. Schrijver. Decomposition of graphs on surfaces and a homotopic circulation theorem. *J. Comb. Theory, Ser. B*, 51(2):161–210, 1991. doi:10.1016/0095-8956(91)90036-J.

40 A. Schrijver. On the uniqueness of kernels. *J. Comb. Theory, Ser. B*, 55(1):146–160, 1992. doi:10.1016/0095-8956(92)90038-Y.

41 Ernst Steinitz. Polyeder und raumeinteilungen. In *Encyclopädie der mathematischen Wissenschaften*, volume Band 3 (Geometries), pages 1–139. B.G. Teubner Verlag, 1922.

42 Thom Sulanke. Generating irreducible triangulations of surfaces, 2006. arXiv:math/0606687.

Locality Sensitive Hashing in Hyperbolic Space

Chengyuan Deng
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Jie Gao
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Kevin Lu
Department of Mathematics, Rutgers University, Piscataway, NJ, USA

Feng Luo
Department of Mathematics, Rutgers University, Piscataway, NJ, USA

Cheng Xin
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Abstract

For a metric space (X, d), a family $\mathcal{H}$ of locality sensitive hash functions is called (r, cr, p_1, p_2) sensitive if a randomly chosen function $h \in \mathcal{H}$ has probability at least p_1 (at most p_2) to map any $a, b \in X$ in the same hash bucket if $d(a, b) \leq r$ (or $d(a, b) \geq cr$). Locality Sensitive Hashing (LSH) is one of the most popular techniques for approximate nearest-neighbor search in high-dimensional spaces, and has been studied extensively for Hamming, Euclidean, and spherical geometries. An (r, cr, p_1, p_2)-sensitive hash function enables approximate nearest neighbor search (i.e., returning a point within distance cr from a query q if there exists a point within distance r from q) with space $O(n^{1+\rho})$ and query time $O(n^{\rho})$ where $\rho = \frac{\log 1/p_1}{\log 1/p_2}$. But LSH for hyperbolic spaces $\mathbb{H}^d$ remains largely unexplored. In this work, we present the first LSH construction native to hyperbolic space. For the hyperbolic plane ($d = 2$), we show a construction achieving $\rho \leq 1/c$, based on the hyperplane rounding scheme. For general hyperbolic spaces ($d \geq 3$), we use dimension reduction from $\mathbb{H}^d$ to $\mathbb{H}^2$ and the 2D hyperbolic LSH to get $\rho \leq 1.59/c$. On the lower bound side, we show that the lower bound on ρ of Euclidean LSH extends to the hyperbolic setting via local isometry, therefore giving $\rho \geq 1/c^2$.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Random projections and metric embeddings; Theory of computation → Nearest neighbor algorithms

Keywords and phrases Locality Sensitive Hashing, Hyperbolic Geometry, Dimension Reduction, Approximate Nearest Neighbor Search

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.39

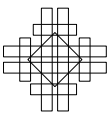
Related Version *Full Version*: `http://arxiv.org/abs/2603.19724` [20]

Funding The authors would like to acknowledge funding support from NSF through CCF-2118953, DMS-2311064, DMS-2220271, IIS-2229876, CNS-2515159.

1 Introduction

Locality Sensitive Hashing (LSH) is a hashing scheme in which similar items are more likely to hash to the same bucket. Indyk and Motwani, in their seminal work [27, 25], developed LSH for general metric spaces and provided hash function constructions for the Hamming distance. The LSH framework provides an efficient data structure for the Approximate Nearest Neighbor Search (ANNS) problem, which is particularly challenging in the high-dimensional regimes. Since then, LSH has been developed for various similarity measures such as ℓ_1 [1, 35], ℓ_2 [18, 2], ℓ_p [18], cosine similarity [15], Jaccard similarity [11, 10],


© Chengyuan Deng, Jie Gao, Kevin Lu, Feng Luo, and Cheng Xin;
licensed under Creative Commons License CC-BY 4.0



angular (spherical) distances [47], etc. With support for similarity search, data independence, mild dependence on data dimension, and simplicity in implementation, LSH serves as an algorithmic tool for classical problems such as clustering, near-duplicate detection, and efficient information retrieval for very large datasets. LSH has also been used widely in database and machine learning pipelines in industry [22, 12, 16, 44, 42, 17, 32, 4, 7]. For a nice survey, please refer to [2].

Hyperbolic geometry is a non-Euclidean geometry. It drops the parallel postulate of Euclidean geometry. In hyperbolic geometry, for any line and a point not on that line, there are at least two distinct lines through the point that do not intersect the original line. Hyperbolic geometry has received increasing attention in modern data analysis because it naturally models hierarchical, tree-like, and exponentially branching structures that arise throughout science and technology. Unlike Euclidean spaces, where volume grows polynomially, hyperbolic spaces exhibit exponential volume growth, matching the combinatorial growth of hierarchies, taxonomies, knowledge graphs, and high-dimensional networks with latent tree-like structure. This geometric compatibility yields embeddings with dramatically reduced distortion: objects that are far apart in a hierarchy can be faithfully separated, and geodesics efficiently capture ancestry, flow, and community structure. As a result, hyperbolic embeddings now underpin state-of-the-art methods in network representation learning, hierarchical clustering, and discrete geometric data analysis [38, 24, 21, 48, 19, 13, 14, 49, 30, 46]. It is also the most important space representing the negative curvature phenomena arising from complex geometric situations [34, 45]. Namely, according to Gromov's hyperbolization program [23], spaces with complicated topology tend to resemble spaces with negative curvature. Hyperbolic geometry provides the fundamental model for negatively curved spaces, offering the canonical framework in which their curvature and geometric behavior are understood.

Moreover, the hyperbolic plane offers an unusually tractable analytic structure: its constant negative curvature provides explicit formulas for distance, geodesics, convexity, and isometries, enabling efficient optimization for machine-learning tasks. Its rich isometry group and uniformization properties are directly linked to complex analysis and conformal representations, providing tools that scale from theoretical guarantees to practical algorithms for dimensionality reduction, visualization, and manifold learning.

These applications provide strong motivation for developing efficient algorithms in hyperbolic space, and in this paper, we study locality-sensitive hashing in hyperbolic space, a topic that, to our best knowledge, has not been studied before.

1.1 LSH Framework

We first recall the formal definitions on LSH and the Approximate Nearest Neighbor Search (ANNS) problem.

▶ **Definition 1** (Locality Sensitive Hashing). *Consider a metric space (X, d)[1], let U be a countable set and $\mathcal{H}$ be a family of functions from X to U. $\mathcal{H}$ is an (r, cr, p_1, p_2)-sensitive LSH family, where $r > 0, c > 1$, $0 < p_2 < p_1 < 1$ are real numbers, if for any two points $x, y \in X$ and a hash function h chosen uniformly at random from $\mathcal{H}$ the following holds*

$$d(x,y) \le r \implies \Pr_{h\sim\mathcal{H}}[h(x) = h(y)] \ge p_1,$$
$$d(x,y) \ge cr \implies \Pr_{h\sim\mathcal{H}}[h(x) = h(y)] \le p_2.$$

[1] LSH is well-defined with a general dissimilarity measure, i.e., the triangle inequality does not necessarily hold. However, the approximate nearest neighbor search problem is typically studied in a metric space. Thus, we define LSH within the same context.

▶ **Definition 2** ((c,r)-approximate Nearest Neighbor Search Problem). *Let P be a point set in metric space (X,d). For any query point $x \in X$, if there exists a point $y \in P$ s.t. $d(x,y) \le r$, then the algorithm returns $y' \in P$ s.t. $d(x,y') \le c \cdot r$.*

Previous works [27, 22] have shown that the approximate nearest neighbor search problem can be solved by LSH, as formally stated below.

▶ **Proposition 3.** *Given an (r, cr, p_1, p_2)-sensitive family $\mathcal{H}$, the (c,r)-ANNS problem can be solved using $O(n^{1+\rho})$ space, and $O(n^{\rho}\log n)$ query time, where $\rho = \rho(\mathcal{H}) = \frac{\log(1/p_1)}{\log(1/p_2)}$.*

Since both the space and time complexity are determined by ρ, we regard ρ as an important performance parameter of LSH families. Namely, we ask for the tight bounds of ρ related to the approximation factor c for a specific metric space.

The tradeoff between the approximation factor c and the performance exponent ρ is intrinsically tied to the geometry of the underlying metric space. Previous studies have established a few tight bounds, with $\rho = \Theta(1/c)$ for Hamming distances and $\rho = \Theta(1/c^2)$ for Euclidean distances [27, 18, 2, 37, 40]. For the unit sphere under angular distance, Charikar [15] proposed LSH based on hyperplane rounding schemes, which are refined by subsequent work [3] showing $\rho = \Theta(1/c^2)$. However, LSH remains underexplored for spaces of negative curvature, such as hyperbolic space. From a theoretical perspective, this gap is notable because hyperbolic geometry exhibits exponential volume expansion, which fundamentally distinguishes itself from the polynomial growth of Euclidean space. Therefore, even extending standard space-partitioning techniques such as quadtree [31] to hyperbolic geometry is non-trivial.

Last, we remark that there has also been work on improved bounds (and empirical performance) for LSH in the data-dependent setting [4, 7, 8, 5, 6, 28]. The focus of this paper is on the data-independent case where the hash function is oblivious to input data P.

1.2 Our Results and Techniques

In this paper, we give the first LSH construction in the hyperbolic space. We start with an LSH design for points in the hyperbolic plane, and then consider LSH families for high-dimensional hyperbolic spaces.

For points in $\mathbb{H}^2$, we are inspired by the hyperplane rounding scheme by Charikar [15] for cosine similarity between vectors. The hyperplane rounding scheme partitions the space by selecting a random hyperplane passing through the origin. Each data point is assigned a bit (0 or 1) determined by the sign of its projection onto the hyperplane's normal vector. In our construction, we partition the space using a random geodesic, which is a circular arc orthogonal to the unit circle in the Poincaré disk model.

A technical caveat here is that we require a well-defined measure on the set of all geodesics. Such a measure is essential to ensure that random sampling of geodesics is meaningful and invariant under hyperbolic isometries. Fortunately, the classical integral geometry provides the needed theory. With some careful crafting using Crofton's formula, we are able to show that our construction yields $\rho \le 1/c$ in $\mathbb{H}^2$. This is formally stated as below.

▶ **Theorem 4.** *For points in the hyperbolic plane $\mathbb{H}^2$ and $r > 0, c > 1$, there exists an (r, cr, p_1, p_2)-sensitive family $\mathcal{H}$, for some $0 < p_2 < p_1 < 1$, such that $\rho(\mathcal{H}) = \frac{\log(1/p_1)}{\log(1/p_2)} \le 1/c$.*

To handle points in high-dimensional hyperbolic space, we use a random projection to low-dimensional hyperbolic space. First, we provide a refined analysis of the Johnson-Lindenstrauss type dimension-reduction result in [9]. Given a point set P in a metric space (X,d), we say a dimension reduction is an (α,β)-approximation to dimension k,

if there exists a mapping f that projects P into a k-dimension space $(\hat{X}, \hat{d})$, such that $\alpha \cdot d(p_i, p_j) \leq \hat{d}(f(p_i), f(p_j)) \leq \beta \cdot d(p_i, p_j)$ for any $p_i, p_j \in P$. In the literature, the embedding distortion is defined as β/α.

▶ **Theorem 5** (Informal Version of Theorem 12). *Let $k \in \mathbb{N}, k \geq 1$, if there exists an (α, β)-approximate dimension reduction for an n-dimension Euclidean point set to k-dimension Euclidean space, then there exists an (α, β)-approximate dimension reduction for a $(n+1)$-dimension hyperbolic point set to $(k+1)$-dimension hyperbolic space.*

Theorem 5 may be of independent interest. For example, an immediate corollary would be a Johnson-Lindenstrauss lemma for hyperbolic space. Namely, for n points in a high-dimensional hyperbolic space, their pairwise distances can be preserved within $(1 \pm \varepsilon)$-approximation while the dimension can be reduced to $O(\log n/\varepsilon^2)$, for $0 < \varepsilon < 1$.

In the context of LSH, our goal is to apply dimension reduction to obtain a projected point set in $\mathbb{H}^2$ and then employ our random geodesic partition technique. By Theorem 5, this task reduces to finding a Euclidean dimension reduction to a straight line ($k = 1$). Interestingly, this has been explored in the first Euclidean LSH construction [18], where they sample a projection matrix from the p-stable distribution (the Gaussian distribution for $p = 2$) and project the input point set onto a straight line. We adopt this scheme and analyze the distance distortion in the hyperbolic plane. Formally, we obtain the following result for a general hyperbolic space.

▶ **Theorem 6.** *For points in d-dimensional hyperbolic space $\mathbb{H}^d$ with $d \geq 3$ and $r > 0, c > 1$, there exists an (r, cr, p_1, p_2)-sensitive family $\mathcal{H}$, for some $0 < p_2 < p_1 < 1$, such that $\rho(\mathcal{H}) = \frac{\log(1/p_1)}{\log(1/p_2)} \leq 1.59/c$.*

The existence of this family immediately implies that a sublinear query time of $\tilde{O}(n^\rho)$ for Hyperbolic ANNS is possible, for $c > 1.59$ in dimension $d \geq 3$. To the best of our knowledge, this is the first worst-case, data-oblivious sublinear query time guarantee for the ANNS problem in high-dimensional hyperbolic space.

We remark that prior work [31] has shown *Locality Sensitive Ordering (LSO)* is not possible in hyperbolic space: no small collection of total orders can make all hyperbolic neighborhoods contiguous. This impossibility stems from the exponential volume growth of $\mathbb{H}^d$, which forces many mutually distant regions to interleave along a global ordering. Therefore, a global geometric linearization is not possible for hyperbolic space. On the other hand, our results confirm that this barrier does not apply to LSH, since hashing only requires distinguishing near and far points with good *probability*, which can be achieved by the randomized geodesic-partition scheme.

On the lower bound side, it is straightforward to argue that the Euclidean LSH lower bound on ρ extends to the hyperbolic setting. This follows from the fact that hyperbolic space is locally Euclidean; by scaling down a "hard" Euclidean dataset to a sufficiently small diameter, it can be embedded into hyperbolic space with arbitrarily low distortion. Therefore, we have the following result for the hyperbolic LSH lower bound.

▶ **Theorem 7.** *Fix $d \in \mathbb{N}$, $c > 1$, then there exists $0 < \tau < 1$, such that any $(\tau d, c\tau d, p_1, p_2)$-sensitive hashing family $\mathcal{H}$ in $\mathbb{H}^d$, for $0 < p_2 < p_1 < 1$, must satisfy $\rho(\mathcal{H}) \geq 1/c^2 - o_d(1)$.*

Our results leave a gap on ρ between $O(1/c)$ and $\Omega(1/c^2)$. Closing this gap remains a major open problem on this topic.

Experiments. We show the performance of the proposed LSH constructions in both the hyperbolic plane and high-dimensional spaces. We observe that the value of ρ for a random point set is a lot lower than $1/c$, which suggests the potential of the hyperbolic LSH in real-world applications.

1.3 Related Work on ANNS in Hyperbolic Spaces

Nearest neighbor query against a given set of n points in the hyperbolic plane can be solved by using a standard point location query in the hyperbolic Voronoi diagram (that can be efficiently constructed [39]). With linear storage, a nearest-neighbor query can be performed in $O(\log n)$ time. This method is limited to 2D hyperbolic space. For n points in $\mathbb{H}^d$ for a constant d, a $(1+\varepsilon)$-approximate nearest neighbor search can be done in time $O(d^{O(d)} \log n \log(1/\varepsilon)/\varepsilon^d)$ with space $O(nd^{O(d)} \log(1/\varepsilon)/\varepsilon^d)$ [31]. The performance deteriorates as d increases. Similar to the Euclidean setting, LSH is typically applied in high-dimensional spaces.

Krauthgamer and Lee [33] studied approximate nearest neighbor in locally doubling, δ-hyperbolic spaces [23]. Specifically, they considered a δ-hyperbolic metric space that has local geometry of type $(\lambda, \delta/3)$ in which every ball of radius $\delta/3$ in X can be covered by λ balls of half the radius. With $O(n^2)$ storage and query time $\lambda^{O(1)} \log^2 n$, an approximate nearest neighbor with an $O(\delta)$ additive error is returned. Note that this result relies on a bounded doubling parameter λ, which is not necessarily true for a general point set in hyperbolic space $\mathbb{H}^d$.

On the practical side, Wu and Charikar [50] used Euclidean nearest neighbor search oracle as a black box for ANNS in hyperbolic spaces. Prokhorenkova et. al. [41] used graph-based nearest neighbor search on a similarity graph and analyzed query time assuming that the points are uniformly distributed within a ball of radius R. Another relevant result is by [5], which shows a data-dependent LSH for general metric space. In their construction, the LSH performance depends on the *cutting modulus* of the metric space.

2 Locality Sensitive Hashing for $\mathbb{H}^2$

In this section, we show the LSH construction in $\mathbb{H}^2$ that yields Theorem 4. We adopt the Poincaré disk model for points in $\mathbb{H}^2$, defined on the unit disk $\mathbb{D} = \{z \in \mathbb{C} : |z| < 1\}$. The space is equipped with the metric $ds = \frac{2|dz|}{1-|z|^2}$, which induces the following distance function:

$$d_{\mathbb{H}}(u, v) = \operatorname{arccosh}\left(1 + 2\frac{|u-v|^2}{(1-|u|^2)(1-|v|^2)}\right).$$

Geometrically, geodesics in $\mathbb{D}$ correspond to circular arcs orthogonal to the unit circle boundary C_∞, with diameters passing through the origin as special cases.

The locality sensitive hash functions we use are *random geodesics* in the Poincaré disk model. For this purpose we require a well-defined probability measure on the set of all geodesics to ensure that random sampling of geodesics is meaningful and invariant under hyperbolic isometries. Luckily, the classical integral geometry [43] provides the needed theory. Integral geometry studies the measures on the spaces of all lines, or totally geodesic planes in Euclidean, hyperbolic, or spherical spaces, such that the measures are invariant under isometries. The fundamental theorems of Poincaré, Cauchy and Crofton show that one can recover basic geometric quantities such as lengths, areas, or volumes of an object by averaging over the set of geodesics intersecting the object.

In the hyperbolic plane, the isometrically invariant probability measure on the space of all geodesics is given by the *kinematic measure* [43]. Take a point q and a non-zero tangent vector v at q in the hyperbolic plane. The polar coordinate of a geodesic γ with respect to (q, v) is a pair (t, θ) where t is the hyperbolic distance from γ to q and θ is the angle of the perpendicular from q to γ with the tangent vector v. The kinematic measure in geodesic polar coordinates (t, θ) is

$$d\mu(t, \theta) = \cosh(t)\, dt\, d\theta.$$

The remarkable feature of the above formula is that it is independent of the choices of q and v. This measure reflects the fact that the volume element in hyperbolic geometry grows exponentially with t.

Since the space of all geodesics in $\mathbb{H}^2$ has infinite total measure, we limit our choices of geodesics in the family of all geodesics intersecting a hyperbolic disk $B(0, R)$ of radius R centered at the origin. By Crofton's formula, the measure of the set of geodesics that intersect a convex body is proportional to the perimeter of that body. Specifically, the corresponding measure is obtained by integrating the invariant element $d\mu$ over the disk:

$$\int_0^{2\pi} \int_0^R \cosh(t)\, dt\, d\theta = 2\pi \sinh(R). \tag{1}$$

Now, we choose R such that the disk $B(0, R)$ centered at the origin with radius R contains all points of P. Our final bound on the performance parameter ρ does not depend on R, and in practice, one can choose R to be a sufficiently large value. Note that in practice, hyperbolic embedding typically avoids using points that are far from the origin due to practical concerns on resolution and numerical instability [36].

A geodesic γ is chosen with probability density function $\frac{\cosh(t)}{2\pi \sinh(R)}$ to be at polar coordinate (t, θ) where t is taken from $[0, R]$ and θ is taken from $[0, 2\pi)$. In the Poincaré disk model, geodesics appear as Euclidean circular arcs, so γ is a circular arc with Euclidean center $c \in \mathbb{R}^2$ and radius $b > 0$. Specifically, define $c = \coth(t)e^{i\theta}$ (here convert $e^{i\theta}$ from the complex plane to the Euclidean plane) and $b = 1/\sinh t$. Define a hash function $\mathbb{H}^2 \to \{-1, 1\}$ by

$$h_\gamma(x) = \operatorname{sgn}\left(\|x - c\|^2 - b^2\right).$$

where x is the Euclidean coordinate of a point in $\mathbb{H}^2$. We can also rewrite the geodesic γ using the Minkowski model with inner product $\langle z, z\rangle = -z_1^2 + z_2^2 + z_3^2$. Let the geodesic γ be given by $\{x \in \mathbb{H}^2 | \langle x, u\rangle = 0\}$ for a point u in the de Sitter space $\{y \in \mathbb{R}^{2,1} | \langle y, y\rangle = 1\}$:

$$u = (\sinh t, \cosh t \cos\theta, \cosh t \sin\theta).$$

Note that (t, θ) is polar coordinate with respect to the point $q = (1, 0, 0)$ and $v = (0, 1, 0)$. Then the formula for $h_\gamma(x)$ is $\operatorname{sgn}\langle x, u\rangle$. This is exactly the same formula as in the Euclidean space. The advantage of this formula over the above one is that it is intrinsic to hyperbolic geometry, i.e., invariant under hyperbolic isometry.

With this setup, we present the hash function in $\mathbb{H}^2$ below.

Input: A point set $P \subset \mathbb{H}^2$

1. Let R be a radius such that P is contained in the ball $B(0, R)$.
2. Let μ_R denote the normalized kinematic measure restricted to the set of geodesics intersecting $B(0, R)$.
3. Sample a random geodesic $\gamma \sim \mu_R$.
4. Hash a point $x \in P$ to $h_\gamma(x) \in \{+1, -1\}$ according to which side of γ it lies on (ties have measure zero).

For brevity, we write $h(x)$ for $h_\gamma(x)$ when it is clear from context. Figure 1 demonstrates how the hash function works as a separator.

The key to proving Theorem 4 is a characterization of the collision probability under our hashing scheme, given in the following lemma.

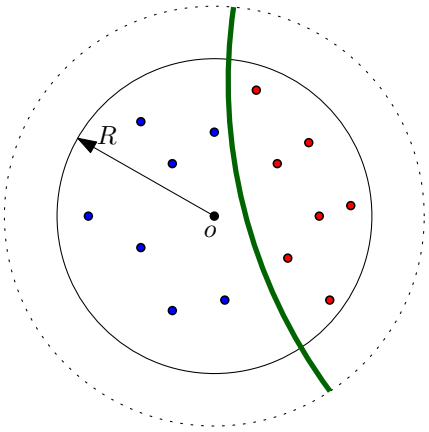


Figure 1 Illustration of the hashing scheme in $\mathbb{H}^2$. If two points are far, then they are more likely to be separated by a random geodesic (the green arc). Points in blue are assigned label $+1$ and points in red are assigned label -1.

▶ **Lemma 8.** *Given two points x, y in $\mathbb{H}^2$ of distance $d_{\mathbb{H}}(x, y) = r$, the kinematic measure μ of the set of all geodesics separating x, y is $2r$. In particular, if $x, y \in B(0, R)$, the ball of radius R is centered at 0, the collision probability with respect to the normalized kinematic measure μ_R is*

$$\Pr[h(x) = h(y) | d_{\mathbb{H}}(x, y) = r] = 1 - \frac{r}{\pi \sinh(R)}. \tag{2}$$

Proof. We use the Poincaré disk model for $\mathbb{H}^2$ in the calculation below. For two points x, y in $\mathbb{H}^2$ of distance $d_{\mathbb{H}}(x, y) = r$, let X be the set of all geodesics in $\mathbb{H}^2$ separating x, y and Y be the set of all geodesics intersecting $B(0, R)$. Note that when $x, y \in B(0, R)$, $X \subset Y$, i.e., $X \cap Y = X$. This shows,

$$\Pr[h(x) \neq h(y) | d_{\mathbb{H}}(x, y) = r] = \frac{\mu(X \cap Y)}{\mu(Y)} = \frac{\mu(X)}{\mu(Y)}. \tag{3}$$

By Equation (1), $\mu(Y) = 2\pi \sinh(R)$. Next we show $\mu(X) = 2r$, where $d_{\mathbb{H}}(x, y) = r$.

By the isometric invariance of the kinematic measure, to compute $\mu(X)$, we may assume, after applying a suitable isometry, that the midpoint of x and y is the origin, and that both points lie on the x-axis. Use the polar coordinate (t, θ) of a random geodesic γ with respect to $(0, x\text{-axis})$. If γ intersects the segment from x to y, the furthest intersection is x or y. Figure 2 illustrates this scenario. By the hyperbolic cosine law, we get

$$\tanh(t) = \tanh(\frac{r}{2}) \cos(\theta).$$

It follows that X in the polar coordinate is

$$X = \{(t, \theta) | \theta \in [-\pi, \pi], 0 \leq t \leq \operatorname{arctanh}(\tanh(\frac{r}{2}) \cos\theta)\}.$$

Therefore,

$$\mu(X) = 2 \int_{-\pi/2}^{\pi/2} \int_{0}^{\operatorname{arctanh}(\tanh(\frac{r}{2}) \cos\theta)} \cosh(t)\, dt\, d\theta.$$

The double integral enjoys a very simple and elegant result, which is r as shown in Claim 9.

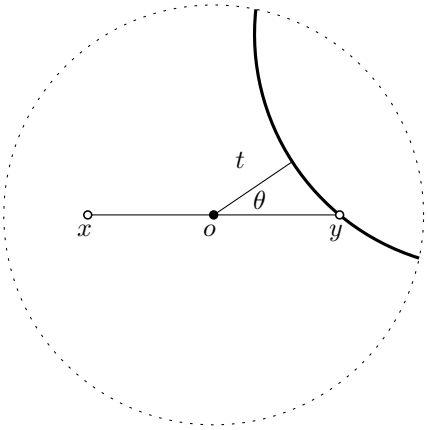


Figure 2 The geodesic intersects y, yielding the maximum value of t.

▷ Claim 9. Let r be a real number.

$$\int_{-\pi/2}^{\pi/2} \int_{0}^{\operatorname{arctanh}(\tanh(r/2)\cos\theta)} \cosh(t)\, dt\, d\theta = r$$

Therefore, $\mu(X) = 2r$. Furthermore, the probability that x, y are separated by γ when γ is chosen from Y is:

$$\Pr[h(x) \neq h(y) | d_{\mathbb{H}}(x,y) = r] = \frac{2r}{2\pi \sinh(R)}$$

and the collision probability is:

$$\Pr[h(x) = h(y) | d_{\mathbb{H}(x,y)} = r] = 1 - \frac{r}{\pi \sinh(R)}$$

This finishes the proof. ◀

The detailed calculation of Claim 9 is deferred to the full version [20]. For now we focus on the upper bound of ρ, note that the lemma immediately gives

$$p_1 = 1 - r/\pi \sinh(R),\ p_2 = 1 - cr/\pi \sinh(R)$$

In order to bound $\rho = \frac{\log 1/p_1}{\log 1/p_2}$, we apply Lemma 1 in [18], copied below.

▶ **Lemma 10** (Lemma 1 [18]). *For $x \in [0,1)$ and $\ell > 1$ such that $1 - \ell x > 0$,*

$$\frac{\log(1-x)}{\log(1-\ell x)} \leq \frac{1}{\ell}.$$

This shows that $\rho \leq 1/c$. Thus we conclude the proof of Theorem 4.

3 Dimension Reduction in Hyperbolic Space

In this section, we prove the dimension reduction result (Theorem 12). We use the Poincaré half-space model of the hyperbolic space $\mathbb{H}^d$. Each point in $\mathbb{H}^d$ is represented by a pair (z, x) with $z \in \mathbb{R}^+$ and $x \in \mathbb{R}^{d-1}$. The hyperbolic distance between two points $p_1 = (z_1, x_1)$ and $p_2 = (z_2, x_2)$ is defined by the following function

$$d_{\mathbb{H}}(p_1, p_2) = \operatorname{arccosh}\left(1 + \frac{\|x_1 - x_2\|^2 + (z_1 - z_2)^2}{2 z_1 z_2}\right),$$

where $\|x_1 - x_2\|$ is the Euclidean distance between x_1, x_2. For a set of n points $P = \{p_i = (z_i, x_i)\} \subseteq \mathbb{H}^d$, we would like to perform dimension reduction to generate $P' = \{p'_i\}$ in $\mathbb{H}^k$ and keep the pairwise distances similar. The method by Benjamini and Makarychev [9] is to perform dimension reduction for the Euclidean point set $X = \{x_i\} \subseteq \mathbb{R}^{d-1}$ by a function f to $X' = \{x'_i\} \subseteq \mathbb{R}^{k-1}$ and define $P' = \{p'_i\}$ with $p'_i = (z_i, x'_i)$ in $\mathbb{H}^k$. Below, we present a refined analysis that yields slightly tighter bounds on the distortion. Essentially, we show that the bound on the distortion of the Euclidean distances of X' (compared to X) carries over to bound the distortion of the hyperbolic distance of P' compared to P.

▶ **Definition 11.** *Let $f : (U, d_U) \to (V, d_V)$ be a map between two metric spaces. If there are positive constants γ_1, γ_2 such that for any $u_1, u_2 \in U$,*

$$\gamma_1 \cdot d_U(u_1, u_2) \leq d_V(f(u_1), f(u_2)) \leq \gamma_2 \cdot d_U(u_1, u_2),$$

we say that the distance stretch is upper bounded by γ_2 and lower bounded by γ_1 and the distortion is γ_2/γ_1.

▶ **Theorem 12.** *Given n points $P = \{p_i = (z_i, x_i)\}$ in $\mathbb{H}^d$. Suppose we have a function $f : \mathbb{R}^{d-1} \to \mathbb{R}^{k-1}$ for n input points $X = \{x_i\}$ such that $\forall x_i, x_j \in X$,*

$$\gamma_1 \cdot \|x_i - x_j\| \leq \|f(x_i) - f(x_j)\| \leq \gamma_2 \cdot \|x_i - x_j\|,$$

with $0 < \gamma_1 \leq 1$ and $\gamma_2 \geq 1$. Then the stretch bounds and distortion of the map

$$g(p) = g((z, x)) = (z, f(x))$$

are the same: $\forall p_i = (z_i, x_i), p_j = (z_j, x_j) \in P$,

$$\gamma_1 \cdot d_{\mathbb{H}}(p_i, p_j) \leq d_{\mathbb{H}}(g(p_i), g(p_j)) \leq \gamma_2 \cdot d_{\mathbb{H}}(p_i, p_j).$$

The proof follows the main idea in [9], but we have slightly improved analysis and bounds. Specifically, the main idea is to analyze the behavior of the function

$$F_{z_1,z_2}(r) = \operatorname{arccosh}\left(1 + \frac{r^2 + (z_1 - z_2)^2}{2z_1 z_2}\right), \tag{4}$$

which is an increasing function of r, when $z_1, z_2 > 0$. For completeness, we include the full proof in the full version [20].

▶ **Lemma 13.** *For $z_1, z_2 > 0$,*
1. *For $\gamma \geq 1$, $F_{z_1,z_2}(\gamma \cdot r) \leq \gamma \cdot F_{z_1,z_2}(r)$.*
2. *For $0 < \gamma' \leq 1$, $F_{z_1,z_2}(\gamma' \cdot r) \geq \gamma' \cdot F_{z_1,z_2}(r)$.*

Proof of Theorem 12. $d_{\mathbb{H}}(g(p_i), g(p_j)) = F_{z_i,z_j}(\|f(x_i) - f(x_j)\|)$. By definition, $\gamma_1\|x_i - x_j\| \leq \|f(x_i) - f(x_j)\| \leq \gamma_2\|x_i - x_j\|$, with $0 < \gamma_1 \leq 1$ and $\gamma_2 \geq 1$. The claim follows from $d_{\mathbb{H}}(g(p_i), g(p_j)) = F_{z_i,z_j}(\|f(x_i) - f(x_j)\|)$, that $F_{z_i,z_j}(x)$ is a monotonically increasing function of x and Lemma 13. ◀

Specifically, one can apply Johnson Lindenstrauss Lemma [29] on the Euclidean coordinates $\{x_i\}$ – by random linear projection to dimension $O(\log n/\varepsilon^2)$ – and arrive at a corresponding Johnson-Lindenstrauss Lemma in hyperbolic geometry.

▶ **Corollary 14** (Johnson Lindenstrauss Transform in $\mathbb{H}^d$)**.** *For n points $P = \{(z_i, x_i)\}$ in $\mathbb{H}^d$ we can project it to points $P' = \{(z_i, x'_i)\}$ in $\mathbb{H}^k$ with $k = \Theta(\log n/\varepsilon^2) + 1$ and stretch upper and lower bounded by $1 \pm \varepsilon$ respectively.*

4 Locality Sensitive Hashing for $\mathbb{H}^d$

A distribution D is called p-stable [51] with $p \geq 0$ if for any n real numbers $v_1, v_2, \cdots, v_n$ and random independent variables $X_1, X_2, \cdots X_n$ with distribution D, $\sum_i v_i X_i$ has the same distribution as the variable $(\sum_i |v_i|^p)^{1/p} X$, where X is a random variable with distribution D. A Gaussian distribution $\mathcal{N}(0,1)$ is 2-stable. This property has been used to build Euclidean LSH [18], to sketch high-dimensional vectors [26], and for many other applications.

Consider n points $P = \{p_i\}$ in $\mathbb{H}^d$. We use random projection to project these points to $P' = \{p'_i\}$ in $\mathbb{H}^2$ using the method in Section 3. In particular, we take a vector $\vec{a}$ with dimension $d-1$ where each entry is independently taken from a Gaussian distribution $\mathcal{N}(0,1)$. For $p_i = (z_i, x_i)$ in the upper half space model, we have $p_i = (z_i, x'_i)$ in $\mathbb{H}^2$, where $x'_i = \vec{a} \cdot x_i$. We then use the LSH mechanism in Section 2 to map these points into buckets. In this section, we analyze the performance of this mechanism.

In our case, the value $\vec{a} \cdot (p - q)$ for any two vectors $p, q \in \mathbb{R}^{d-1}$ is also a Gaussian distribution with zero mean and variance $\|p - q\|^2$, or, the distribution of $\|p - q\| Z$ with $Z \sim \mathcal{N}(0,1)$. Take $f(z)$ to be the probability density function of the absolute value of $\mathcal{N}(0,1)$:

$$f(z) = \frac{2}{\sqrt{2\pi}} e^{-z^2/2}.$$

To calculate the probability that p_1, p_2 are mapped to the same bucket, we take $s = \|x_1 - x_2\|$ and $r = d_{\mathbb{H}}(p_1, p_2) = F_{z_1,z_2}(s)$, where

$$F_{z_1,z_2}(s) = \operatorname{arccosh}\left(1 + \frac{s^2 + (z_1 - z_2)^2}{2 z_1 z_2}\right),$$

is an increasing function of s. Suppose p_1, p_2 are mapped to points p'_1, p'_2 with $s' = \|x'_1 - x'_2\|$ and $r' = d_{\mathbb{H}}(p'_1, p'_2) = F_{z_1,z_2}(s')$. s' follows the distribution of sZ with $Z \sim \mathcal{N}(0,1)$. Recall that using the LSH for $\mathbb{H}^2$ the probability that p'_1, p'_2 map to the same bucket is $1 - \frac{r'}{w}$ where $w = 2 \sinh R$. Now we have,

$$p(r) = \Pr(h(p_1) = h(p_2)) = \int_{z=0}^{\infty} f(z) \left(1 - \frac{F_{z_1,z_2}(s \cdot z)}{w}\right) dz.$$

We can get an upper bound of $p(r)$.

▶ **Lemma 15.**

$$p(r) \leq 1 - \alpha \frac{r}{w}, \ \alpha \approx 0.63. \tag{5}$$

Proof. We first rewrite:

$$\int_{z=0}^{\infty} f(z) \left(1 - \frac{F_{z_1,z_2}(s \cdot z)}{w}\right) dz = 1 - \int_{z=0}^{1} f(z) \frac{F_{z_1,z_2}(s \cdot z)}{w} dz - \int_{z=1}^{\infty} f(z) \frac{F_{z_1,z_2}(s \cdot z)}{w} dz$$

For the first integral, by Lemma 13, $F_{z_1,z_2}(s \cdot z) \geq z \cdot F_{z_1,z_2}(s) = zr$ in the range of $0 < z \leq 1$.

$$\int_{z=0}^{1} f(z) \frac{F_{z_1,z_2}(s \cdot z)}{w} dz \geq \int_{z=0}^{1} f(z) z \frac{r}{w} dz = \sqrt{\frac{2}{\pi}} (1 - e^{-1/2}) \frac{r}{w} \approx .31 \frac{r}{w}$$

For the second integral, $z \geq 1$, $F_{z_1,z_2}(s \cdot z) \geq F_{z_1,z_2}(s) = r$.

$$\int_{z=1}^{\infty} f(z) \frac{F_{z_1,z_2}(s \cdot z)}{w} dz \geq \int_{z=1}^{\infty} f(z) \frac{r}{w} dz \approx .32 \frac{r}{w}$$

Adding them together gives us:

$$p(z) \leq 1 - .63\frac{r}{w}$$ ◀

We also have a lower bound of $p(r)$.

▶ **Lemma 16.**

$$p(r) \geq 1 - \frac{r}{w}. \tag{6}$$

Proof. Here, z follows the distribution of the absolute value of a Gaussian distribution $\mathcal{N}(0,1)$, $z \geq 0$, $\mathbb{E}[z] = \sqrt{2/\pi}$ and $\mathbb{E}[z^2] = 1$.

The function $\operatorname{arccosh}(x)$ with $x > 1$ is concave and monotonically increasing. For a concave function $b(x)$ one can write it as

$$b(x) \leq b(x_0) + b'(x_0)(x - x_0).$$

Applying this for $b(x) = \operatorname{arccosh}(x)$ at $x = 1 + \frac{s^2 z^2 + (z_1 - z_2)^2}{2z_1 z_2}$ and $x_0 = 1 + \frac{s^2 + (z_1 - z_2)^2}{2z_1 z_2}$ we have

$$F_{z_1,z_2}(sz) \leq F_{z_1,z_2}(s) + b'(x_0) \cdot \frac{s^2(z^2 - 1)}{2z_1 z_2}$$

Taking expectation over z on both sides and recall $\mathbb{E}[z^2] = 1$ we have

$$\mathbb{E}[F_{z_1,z_2}(sz)] \leq F_{z_1,z_2}(s) + b'(x_0) \cdot \frac{s^2(\mathbb{E}[z^2] - 1)}{2z_1 z_2} = F_{z_1,z_2}(s) = r$$

Now we have

$$p(r) = 1 - \mathbb{E}[F_{z_1,z_2}(sz)]/w \geq 1 - r/w$$

This finishes the proof. ◀

Thus $p_1 = p(r) \geq 1 - r/w$. $p_2 = p(c \cdot r) \leq 1 - \frac{\alpha c r}{w}$. Since we need $p_1 > p_2$, this requires that $c \geq 1/\alpha \approx 1.59$.

Now we can estimate the function $\rho = \frac{\log 1/p(r)}{\log 1/p(c \cdot r)}$. Again we apply Lemma 10 and get

$$\rho = \frac{\log 1/p(r)}{\log 1/p(c \cdot r)} \leq \frac{\log(1 - \frac{r}{w})}{\log(1 - \frac{\alpha c r}{w})} \leq \frac{1}{\alpha c} \approx \frac{1.59}{c}.$$

This concludes Theorem 6.

5 Lower Bound

The lower bound on the performance parameter ρ for LSH of ℓ_p metrics has been studied in [37, 40]. Theorem 5.2 in [40] stated that for points of the Hamming cube $\{0,1\}^d$ and some $0 < \delta < 1$, an (r, R, p_1, p_2)-sensitive LSH of Hamming distance, with $r = \delta d/c$ and $R = \delta d$, must have

$$\rho(c) = \frac{\log 1/p_1}{\log 1/p_2} \geq \frac{1}{c} - o_d(1)$$

This lower bound can be used to generate a lower bound of $1/c^p$ for ℓ_p distance – by using ℓ_p distance on the same point set. In particular, it shows that LSH for Euclidean distances has $\rho \geq 1/c^2$. We will use the same construction to show a lower bound on ρ for hyperbolic distance.

▶ **Theorem 7.** *Fix $d \in \mathbb{N}$, $c > 1$, then there exists $0 < \tau < 1$, such that any $(\tau d, c\tau d, p_1, p_2)$-sensitive hashing family $\mathcal{H}$ in $\mathbb{H}^d$, for $0 < p_2 < p_1 < 1$, must satisfy $\rho(\mathcal{H}) \geq 1/c^2 - o_d(1)$.*

Proof. For a point x_i of the Hamming cube $\{0,1\}^d$, we define $p_i \in \mathbb{H}^{d+1}$ to be (z, x_i) in the half-space model, with a very large value of z to be decided later. Now consider the hyperbolic distance of p_i, p_j:

$$d_{\mathbb{H}}(p_i, p_j) = \operatorname{arccosh}\left(1 + \frac{\|x_i - x_j\|^2}{2z^2}\right)$$

For any $0 < \varepsilon < 1$, we take $z \geq \sqrt{d/2\varepsilon}$. Thus $\frac{\|x_i - x_j\|^2}{2z^2} \leq \varepsilon$. Use the generalized power series (Puiseux series)

$$\operatorname{arccosh}(1+x) = \sqrt{2}\sum_{t=0}^{\infty} \frac{(-1)^t \binom{2t}{t}}{(2t+1)8^t} x^{t+1/2} = \sqrt{2x}\left(1 - \frac{1}{12}x + \frac{3}{160}x^2 - \frac{5}{896}x^3 + \cdots\right)$$

This gives us, for small positive x,

$$\sqrt{2x}(1 - \frac{1}{12}x) \leq \operatorname{arccosh}(1+x) \leq \sqrt{2x}.$$

We can now bound

$$\sqrt{\|x_i - x_j\|_1}(1 - \varepsilon/12) \leq d_{\mathbb{H}}(p_i, p_j) \cdot z \leq \sqrt{\|x_i - x_j\|_1},$$

where the ℓ_1 distance (same as the Hamming distance) $\|x_i - x_j\|_1$ is the same as the squared ℓ_2 distance $\|x_i - x_j\|_2^2$. Now we build an (r_h, R_h, p_1, p_2)-sensitive LSH such that $r_h \cdot z \leq \sqrt{r}$ and $\sqrt{R}(1 - \varepsilon/12) \leq R_h \cdot z$. Take $R_h = c_h \cdot r_h$, we have

$$\frac{1}{c} = \frac{r}{R} \geq \left(\frac{r_h}{R_h}\right)^2 (1 - \varepsilon/12)^2 = \frac{1}{c_h^2}(1 - \varepsilon/12)^2$$

Combining everything, a LSH for $\mathbb{H}^{d+1}$ must have

$$\rho(c_h) \geq \frac{1}{c} - o_d(1) \geq \frac{1}{c_h^2}(1 - \varepsilon/12)^2 - o_d(1) = \Omega(1/c_h^2)$$ ◀

6 Experiments

We implement our Locality Sensitive Hashing algorithms in both $\mathbb{H}^2$ and $\mathbb{H}^d$. We aim to evaluate the LSH performance in terms of ρ in practice.

We first explain the synthetic data we use. For each $d \in \{2, 10, 100, 1000\}$, we generate a dataset of 1000 points by uniformly random sampling in a hyperbolic sphere of the same radius, $R = \log(199)$. This radius was chosen so that after the points are projected to $\mathbb{H}^2$ and mapped to the Poincare disk model, the points lie inside a Euclidean circle of radius 0.99. Further, we set $r = 0.2$ with varying $c = 1.5 + k$ for $k = 0, 1...17$ to compute the value of p_1 and p_2. We illustrate results on p_1, p_2 in Figure 3 and ρ in Figure 4 for different choices of c obtained by averaging over 1000 repetitions.

We see that, as expected, p_1 and p_2 are close to 1 in all cases. Due to the randomness of the data, the probabilities vary slightly in percentage. Since ρ is related to the complements of these probabilities, varying slightly in proportion when so close to 1 causes the significant differences in ρ values that we are observing. However, in all cases, the ρ graphs are below

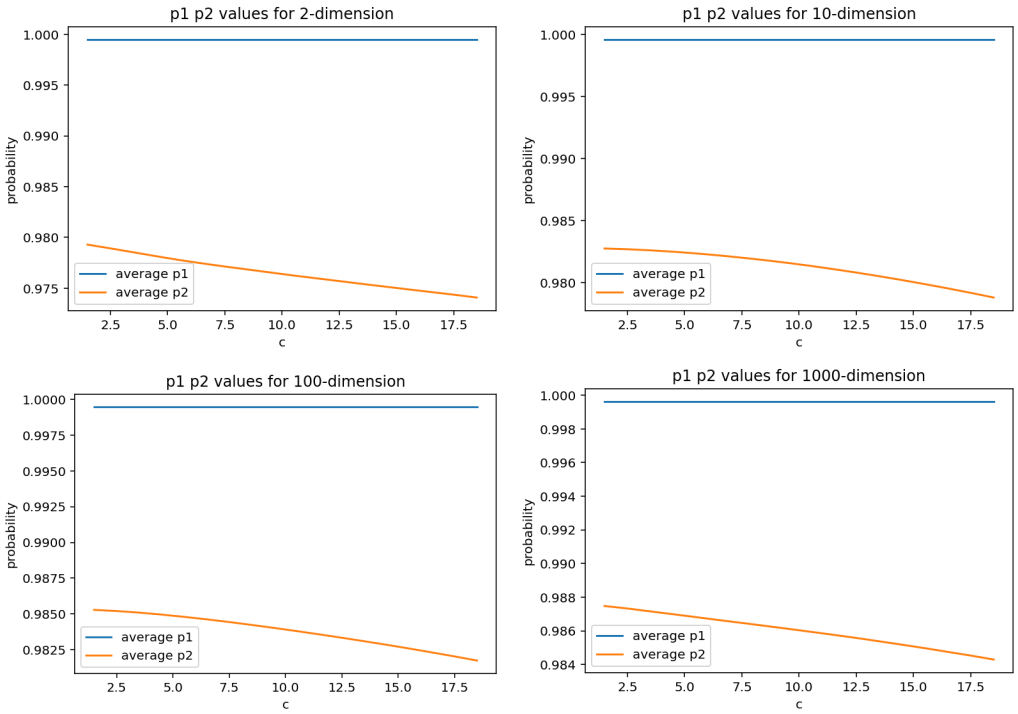


Figure 3 Average p_1, p_2 of Hyperbolic LSH with varying c for different dimensions (y-values do not start at 0).

$1/c$. We note that they are not exactly $1/c$ because we considered all pairs of points with distance below r or above cr, not exactly equal to r or cr, so $1/c$ should be an upper bound of what we obtain.

Next, we use the same data for $d = 2$, but vary $c = 1 + .1k$ for $k = 1, ...10$ to see if the LSH still works for small values of c even though our proof only has guarantees for $c \geq 1.59$. From Figure 5 we see fairly similar results, just p_2 varying less, but LSH still achieves ρ lower than $\frac{1}{c}$.

We also generated some data for $d = 1000$, varying R so that the points lie in a circle of Euclidean radii .9 and .999. We see that p_1, p_2 and ρ change a little, but not significantly and still fall well within the theoretical bounds.

Lastly, we generate similar data for $d = 2$, but containing 2000 points. Then, we analyze p_1 and p_2 for pairs of points that are at the boundary of acceptable distances. Precisely, we choose pairs of points that have distance within $[.9r, r]$ or $[cr, 1.1cr]$. This gives us an approximately linear graph for p_2 and ρ curve that approximates $\frac{1}{c}$.

The experiment setup with a fixed r and a reasonably large constant c (rather than $1 + \varepsilon$ for a very small ε) is motivated by real world applications such as similarity learning which focuses on differentiating data elements that are very similar or sufficiently far away. Many data modalities and data tasks fall into this category. For example, in image recognition the goal is to recognize two pictures with similar scenes or semantics. But for two pictures on completely different subjects, it is sufficient to know that they are far away and the exact values of dissimilarities are less important. One important take-away message from our experiments is that the empirical value of ρ with the synthetic data set is very small with a mild dependency on dimension and decreasing value of c. It is much smaller than

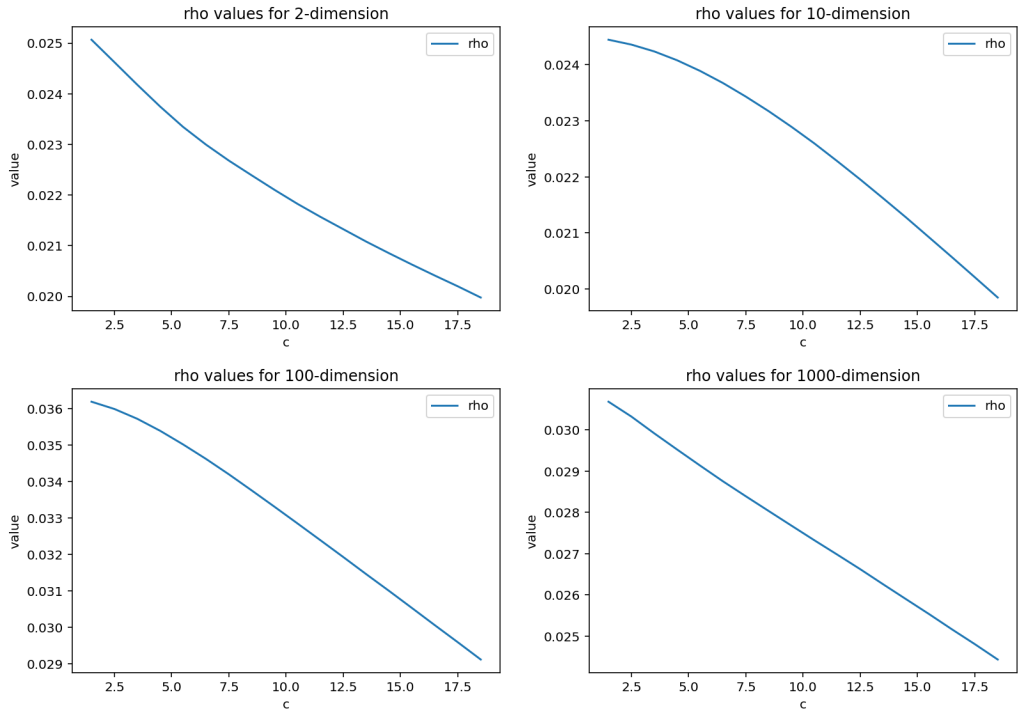


Figure 4 Average ρ of Hyperbolic LSH with varying c for different dimensions (y-values do not start at 0).

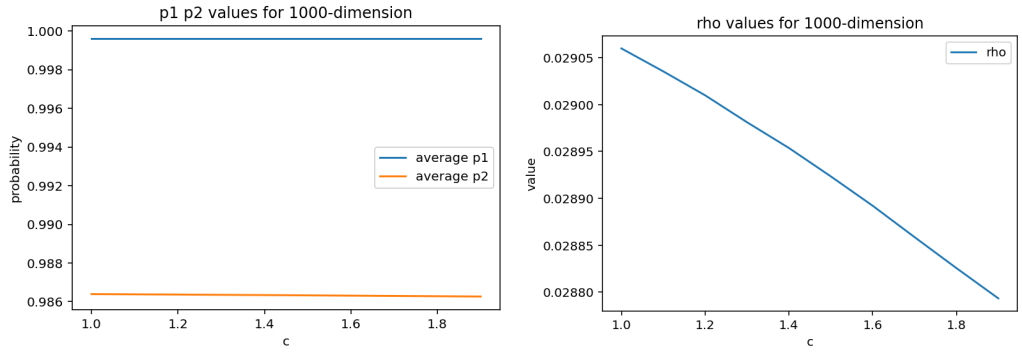


Figure 5 Average ρ and average p_1, p_2 in high-dimension when c is small (y-values do not start at 0).

our theoretical upper bound – partly because we consider all pairs whose distance are often much smaller than r or much greater than cr. Since the storage requirement for approximate nearest neighbor data structure built with LSH grows in terms of $O(n^{1+\rho})$ and the query cost grows in terms of $O(n^{\rho})$. The empirical small value of ρ suggests great potential for practical use.

7 Conclusion

To our knowledge, this is the first work for locality sensitive hashing in hyperbolic space. Our construction achieves a performance parameter $\rho = O(1/c)$ for general hyperbolic space. To complement, we show there is a lower bound of $\rho \approx \Omega(1/c^2)$. The major open question is whether there can be improvement with a better ρ or show a stronger lower bound for

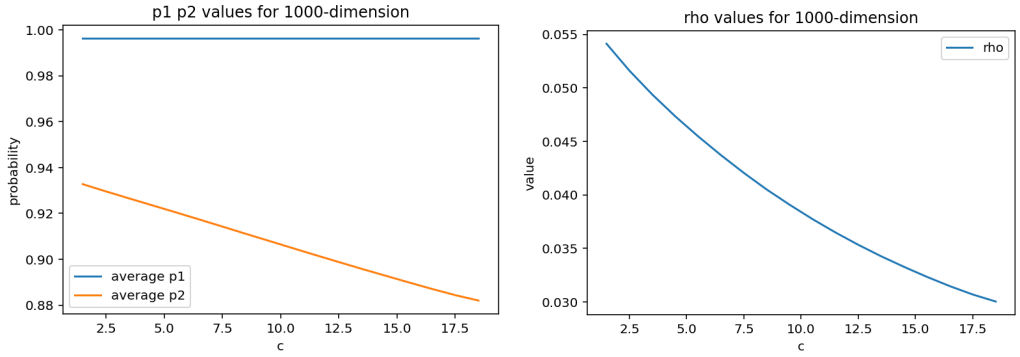


Figure 6 Average ρ and average p_1, p_2 in high dimension and $R = .9$ in Euclidean coordinates (y-values do not start at 0).

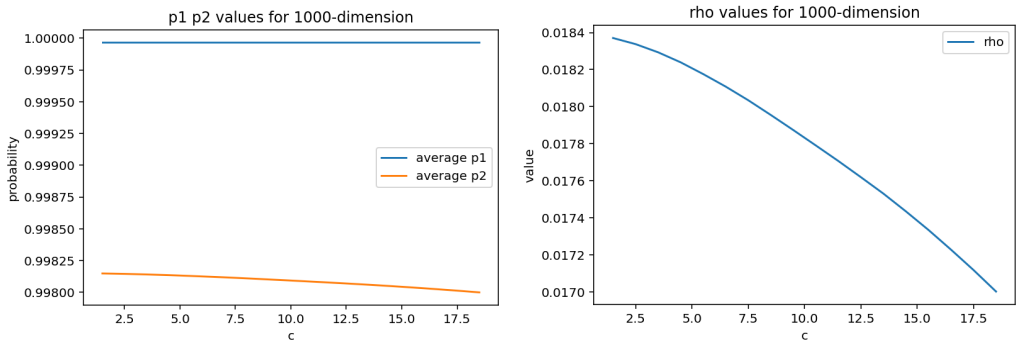


Figure 7 Average ρ and average p_1, p_2 in high dimension and $R = .999$ in Euclidean coordinates (y-values do not start at 0).

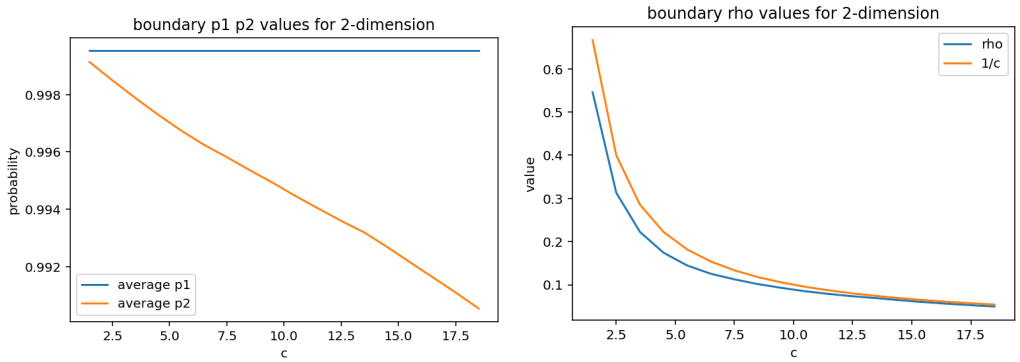


Figure 8 Average ρ and average p_1, p_2 for only pairs at distance approximately r or cr (y-values do not start at 0).

the LSH of hyperbolic distances. Note that prior results are obtained from Boolean analysis that give lower bounds on the Hamming space, which implies a lower bound for ℓ_p spaces. We believe novel techniques are required to show a non-trivial lower bound for hyperbolic space. Another open direction is data-dependent LSH, which outperforms classical LSH in Euclidean and Hamming space. It would be interesting to see if this result extends to hyperbolic space.

References

1 Alexandr Andoni and Piotr Indyk. Efficient algorithms for substring near neighbor problem. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm - SODA '06*, pages 1203–1212, New York, New York, USA, 2006. ACM Press. URL: http://dl.acm.org/citation.cfm?id=1109557.1109690.

2 Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008. doi:10.1145/1327452.1327494.

3 Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 1225–1233. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/2823f4797102ce1a1aec05359cc16dd9-Paper.pdf.

4 Alexandr Andoni, Piotr Indyk, Huy L Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1028. Society for Industrial and Applied Mathematics, Philadelphia, PA, January 2014.

5 Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Data-dependent hashing via nonlinear spectral gaps. In *Proceedings of the 50th annual ACM SIGACT symposium on theory of computing*, pages 787–800, 2018. doi:10.1145/3188745.3188846.

6 Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 159–169. IEEE, 2018. doi:10.1109/FOCS.2018.00024.

7 Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 793–801, New York, NY, USA, June 2015. ACM. doi:10.1145/2746539.2746553.

8 Alexandr Andoni and Ilya Razensteyn. Tight Lower Bounds for Data-Dependent Locality-Sensitive Hashing. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:11, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2016.9.

9 Itai Benjamini and Yury Makarychev. Dimension reduction for hyperbolic space. *Proceedings of the American Mathematical Society*, pages 695–698, 2009.

10 Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29. IEEE Comput. Soc, 2002.

11 Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Comput. Netw. ISDN Syst.*, 29(8-13):1157–1166, 1997. doi:10.1016/S0169-7552(97)00031-7.

12 Jeremy Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428, 2001. doi:10.1093/BIOINFORMATICS/17.5.419.

13 Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914, Stroudsburg, PA, USA, 2020. Association for Computational Linguistics. doi:10.18653/V1/2020.ACL-MAIN.617.

14 Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Adv. Neural Inf. Process. Syst.*, pages 4869–4880, 2019.

15 Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002. doi:10.1145/509907.509965.

16 Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D Ullman, and Cheng Yang. Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):64–78, 2002. doi:10.1109/69.908981.

17 Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280, 2007.

18 Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004. doi:10.1145/997817.997857.

19 Christopher De Sa, Albert Gu, Christopher Ré, and Frederic Sala. Representation tradeoffs for hyperbolic embeddings. *Proc. Mach. Learn. Res.*, pages 4460–4469, 2018.

20 Chengyuan Deng, Jie Gao, Kevin Lu, Feng Luo, and Cheng Xin. Locality sensitive hashing in hyperbolic space, 2026. arXiv:2603.19724.

21 Octavian Ganea, Gary Becigneul, and Thomas Hofmann. Hyperbolic neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 5350–5360. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/dbab2adc8f9d078009ee3fa810bea142-Paper.pdf.

22 Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. URL: http://www.vldb.org/conf/1999/P49.pdf.

23 M. Gromov. Hyperbolic groups. In *Essays in group theory*, volume 8 of *Math. Sci. Res. Inst. Publ.*, pages 75–263. Springer, New York, 1987. doi:10.1007/978-1-4613-9586-7_3.

24 Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. Hyperbolic attention networks. In *International Conference on Learning Representations*, 2019. URL: https://openreview.net/forum?id=rJxHsjRqFQ.

25 Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(14):321–350, 2012. doi:10.4086/toc.2012.v008a014.

26 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, pages 307–323, 2006. doi:10.1145/1147954.1147955.

27 Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998. doi:10.1145/276698.276876.

28 Rajesh Jayaram, Erik Waingarten, and Tian Zhang. Data-dependent LSH for the earth mover's distance. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 800–811, New York, NY, USA, June 2024. ACM. doi:10.1145/3618260.3649666.

29 William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

30 Baki Karliğa. Edge matrix of hyperbolic simplices. *Geom. Dedicata*, 109(1):1–6, December 2004.

31 Sándor Kisfaludi-Bak and Geert van Wordragen. A Quadtree, a Steiner spanner, and approximate nearest neighbours in Hyperbolic space. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.68.

32 Hisashi Koga, Tetsuo Ishibashi, and Toshinori Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowl. Inf. Syst.*, 12(1):25–53, May 2007. doi:10.1007/S10115-006-0027-5.

33 Robert Krauthgamer and James Lee. Algorithms on negatively curved spaces. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 119–132. IEEE, 2006. doi:10.1109/FOCS.2006.9.

34 Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, September 2010. doi:10.1103/PhysRevE.82.036106.

35 Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 577–591. IEEE Comput. Soc. Press, 2002.

36 Gal Mishne, Zhengchao Wan, Yusu Wang, and Sheng Yang. The numerical stability of hyperbolic representation learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24925–24949. PMLR, 23–29 July 2023. URL: https://proceedings.mlr.press/v202/mishne23a.html.

37 Rajeev Motwani, Assaf Naor, and Rina Panigrahi. Lower bounds on locality sensitive hashing. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 154–157, 2006. doi:10.1145/1137856.1137881.

38 Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6341–6350, Red Hook, NY, USA, 2017. Curran Associates Inc.

39 Frank Nielsen and Richard Nock. Hyperbolic voronoi diagrams made easy. In *2010 International Conference on Computational Science and Its Applications*. IEEE, 2010.

40 Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *ACM Transactions on Computation Theory (TOCT)*, 6(1):1–13, 2014. doi:10.1145/2578221.

41 Liudmila Prokhorenkova, Dmitry Baranchuk, Nikolay Bogachev, Yury Demidovich, and Alexander Kolpakov. Graph-based nearest neighbor search in hyperbolic spaces. In *International Conference on Learning Representations*, 2022. URL: https://openreview.net/forum?id=USIgIY6TNDe.

42 Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. Randomized algorithms and NLP: Using locality sensitive hash functions for high speed noun clustering. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 622–629, 2005. doi:10.3115/1219840.1219917.

43 Luis A Santalo and Mark Kac. *Integral geometry and geometric probability*. Cambridge University Press, Cambridge, October 2004.

44 Shakhnarovich, Viola, and Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 750–757. IEEE, 2003.

45 Yuval Shavitt and Tomer Tankel. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *IEEE INFOCOM 2004*, volume 1, page 384. IEEE, 2004. doi:10.1109/INFCOM.2004.1354510.

46 Puoya Tabaghi, Michael Khanzadeh, Yusu Wang, and Siavash Mirarab. Principal component analysis in space forms. *IEEE Trans. Signal Process.*, 72:4428–4443, 2024. doi:10.1109/TSP.2024.3457529.

47 Kengo Terasawa and Yuzuru Tanaka. Spherical LSH for approximate nearest neighbor search on unit hypersphere. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 27–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-73951-7_4.

48 Alexandru Tifrea, Gary Becigneul, and Octavian-Eugen Ganea. Poincaré glove: Hyperbolic word embeddings. In *International Conference on Learning Representations*, 2019. URL: https://openreview.net/forum?id=Ske5r3AqK7.

49 Richard C Wilson, Edwin R Hancock, Elzbieta Pekalska, and Robert P W Duin. Spherical and hyperbolic embeddings of data. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 2255–2269, 2014. doi:10.1109/TPAMI.2014.2316836.

50 Xian Wu and Moses Charikar. Nearest neighbor search for hyperbolic embeddings. *arXiv [cs.DS]*, 2020. arXiv:2009.00836.

51 V. M. Zolotarev. *One-Dimensional Stable Distributions*, volume 65 of *Translations of Mathematical Monographs*. American Mathematical Society, 1986.

Computing the Intrinsic Delaunay Triangulation of a Closed Polyhedral Surface

Loïc Dubois[1]
Notre Dame, IN, USA

Abstract

Every surface that is intrinsically polyhedral can be represented by a portalgon: a collection of polygons in the Euclidean plane with some pairs of equally long edges abstractly identified. While this representation is arguably simpler than meshes (flat polygons in $\mathbb{R}^3$ forming a surface), it has unbounded *happiness*: a shortest path in the surface may visit the same polygon arbitrarily many times. This pathological behavior is an obstacle towards efficient algorithms. On the other hand, Löffler, Ophelders, Staals, and Silveira [SoCG 2023] recently proved that the (intrinsic) Delaunay triangulations have bounded happiness.

In this paper, given a closed polyhedral surface S, represented by a triangular portalgon T, we provide an algorithm to compute the Delaunay triangulation of S whose vertices are the singularities of S (the points whose surrounding angle is distinct from 2π). The time complexity of our algorithm is polynomial in the number of triangles and in the logarithm of the aspect ratio r of T. Within our model of computation, we show that the dependency in $\log r$ is unavoidable. Our algorithm can be used to pre-process a triangular portalgon before computing shortest paths on its surface, and to determine whether the surfaces of two triangular portalgons are isometric.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Polyhedral surface, intrinsic Delaunay triangulation, algorithmic complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.40

Related Version *Full Version*: https://arxiv.org/abs/2601.03954

1 Introduction

In one of its simplest forms a *triangulation* is a finite collection of disjoint triangles in the Euclidean plane, together with a partial matching of the sides of the triangles such that any two matched sides have the same length (Figure 1). This simple representation appears under different names in the literature (intrinsic triangulation [26, 27], portalgon [20]). Cutting out the triangles from the plane and identifying the matched sides isometrically, respecting the orientations of the triangles, provides a (compact, orientable) *polyhedral* surface. This surface is *closed* if, in addition, it is connected and without boundary.

In this paper we consider the *Delaunay* triangulation of a closed polyhedral surface whose vertex set consists of the singularities (the points surrounded by an angle distinct from 2π) of the surface (except for flat tori, see below). It is generically unique. Our main contribution is an algorithm to compute it from any triangulation of the surface, whose time complexity is polynomial in the number of triangles and in the logarithm of the aspect ratio of the input triangulation. Our second contribution is a lower bound showing that the dependency in the logarithm of the aspect ratio is unavoidable in our model of computation.

Before describing our contributions in more detail, we discuss related works.

[1] This work was done while the author was working at LIGM, CNRS, Univ Gustave Eiffel, F-77454 Marne-la-Vallée, France.

© Loïc Dubois;
licensed under Creative Commons License CC-BY 4.0

1.1 Related works

Polyhedral surfaces can also be obtained from *meshes*, flat triangles in $\mathbb{R}^3$ glued along their edges. Moreover, every mesh defines a triangulation of its surface. Yet triangulations are more general than meshes: most triangulations cannot be obtained from a mesh. Some recent algorithms advantageously operate on triangulations of polyhedral surfaces without reference to a mesh [25, 31, 17]. In this context the adjective "intrinsic" is sometimes placed before the name "triangulation" to make the distinction with the particular triangulations arising from a mesh. In the mathematical community, a prominent example is that of a translation surface [21, 32, 14, 12], which arises naturally in the study of billiards in rational polygons.

Triangulations are so general that not all of them are suitable for computation, compared to meshes. Prominently, a fundamental problem on polyhedral surfaces is to compute the distance or a report a shortest path between two points. On a mesh, shortest paths can be computed in time polynomial in the number of triangles. For example, an algorithm of Mitchell, Mount, and Papadimitriou [22] propagates waves along the surface, starting from the source, in a discrete manner. See also Chen and Han [4]. On a generic triangulation however (not arising from a mesh), the number of times a shortest path visits a triangle is not bounded by any function of the number of triangles, as noted for example almost 20 years ago by Erickson [8]. Recently, Löffler, Ophelders, Staals, and Silveira [20] coined the term *happiness* of a triangulation, for the maximum number of times a shortest path visits a triangle. They adapted the single-source shortest paths algorithm of Mitchell, Mount, and Papadimitriou [22] from meshes to triangulations, whose time complexity now depends on the happiness of the triangulation (it is more efficient on triangulations of *low* happiness).

This raises the problem of replacing any given triangulation by another triangulation of the same surface whose happiness is "low". Among the many remeshing algorithms [13, 28, 24, 29], only few have been ported to the general context of intrinsic triangulations [26], and the only solution we are aware of that compares to our main result (Theorem 1 below), by Löffler, Ophelders, Staals, and Silveira [20, Section 5], is restricted to particular inputs whose surfaces are all homeomorphic to an annulus; we will use it as a black box. Importantly, the same authors also showed that Delaunay triangulations have bounded happiness [20, Section 4.2].

Delaunay triangulations are classical objects of computational geometry [5, 11, 2], closely related to shortest paths. While mostly known in the plane, they generalize to closed polyhedral surfaces, see for example the depiction of Bobenko and Springborn [3]. To compute a Delaunay triangulation from an arbitrary intrinsic triangulation there are, to our knowledge, only two approaches, and neither compares to our main result (Theorem 1). One approach computes a Voronoi diagram with a suitably adapted multiple-source shortest path algorithm, and then derives from it a Delaunay tessellation, see for example Mount [23], Liu, Chen, and Tang [18], and Liu, Xu, Fan, and He [19]. Another approach starts from an initial triangulation and flips its edges until it reaches a Delaunay triangulation; it was proved to terminate by Indermitte, Liebling, Troyanov, and Clémençon [3, 15].

1.2 Our results

In order to state our results precisely, it now matters to make the distinction between a triangulation and the data structure representing it, and to allow for more general polygons than triangles. Following Löffler, Ophelders, Staals, and Silveira [20], we call *portalgon* the collection T of polygons in the Euclidean plane and the partial matching of their sides. We denote by $\mathcal{S}(T)$ the associated polyhedral surface. We say that the portalgon T is *triangular* if all polygons are triangles. The sides of the polygons, once identified, constitute a graph T^1 embedded on $\mathcal{S}(T)$ (the polygons themselves correspond to the faces of T^1): it is this graph T^1 that we call *triangulation* if T is triangular, and we call T^1 a *tessellation* in general.

We consider, on a closed polyhedral surface S, the unique Delaunay tessellation $\mathcal{D}$ of S whose vertices are exactly the singularities of S, with a single very special exception: if S has no singularity, then S is a flat torus and we let $\mathcal{D}$ be any of the Delaunay tessellations of S that have exactly one vertex, for one can be mapped to the other via an orientation-preserving isometry of S anyway. In any case, we say that $\mathcal{D}$ is *the* Delaunay tessellation of S, in a slight abuse.[2] It is "generically" a triangulation, but not always. If not, then triangulating the faces of $\mathcal{D}$ along any vertex-to-vertex arcs provides a Delaunay triangulation. The *aspect ratio* of a triangular portalgon T is the maximum side length of a triangle of T divided by the smallest height of a triangle of T (possibly another triangle). Our main contribution is:

▶ **Theorem 1.** *Let T be a portalgon of n triangles, of aspect ratio r, whose surface $\mathcal{S}(T)$ is closed. One can compute the portalgon of the Delaunay tessellation of $\mathcal{S}(T)$ in $O(n^3 \log^2(n) \cdot \log^4(r))$ time.*

As already mentioned, the only two methods we are aware of for computing a Delaunay tessellation from an arbitrary triangulation are the flip algorithm and the computation of the dual Voronoi diagram. The time complexities of these algorithms are not bounded by any polynomial in n and $\log(r)$.

Applications of Theorem 1 are detailed in the full version of the paper. Briefly, on the portalgon returned by Theorem 1, shortest paths can be computed in $O(n^2 \log^{O(1)} n)$ time. And Theorem 1 enables to test whether the surfaces of two given portalgons are isometric, simply by computing and comparing the portalgons of the associated Delaunay tessellations.

We analyze our algorithms within the real RAM model of computation described by Erickson, van der Hoog, and Miltzow [10]. It is an extension of the standard integer word RAM, with an additional memory array storing reals, and with additional instructions. On such a machine, we represent each polygon of a portalgon T by the list of its vertices, and each vertex is by its two coordinates, stored in the memory array dedicated to reals. So displacing (translating or rotating) the polygons in the plane provides different representations of T. When modifying a portalgon T, we actually modify our representation of T, using elementary operations that are easily seen to be achievable by a real RAM.

Within this model of computation, our second contribution is a lower bound that backs our main result, Theorem 1, by showing that the polynomial dependency in the logarithm of the aspect ratio is unavoidable:

▶ **Theorem 2.** *Let $c \in (0, 1)$. There are a flat torus S, and for every $x \in (1, \infty)$, a representation of a portalgon T_x, with two triangles, whose aspect ratio is $O(x^2)$, whose surface is S, that satisfy the following. There is no real RAM algorithm computing a representation of the portalgon of the Delaunay tessellation of S from T_x in $O((\log x)^c)$ time.*

Altogether Theorem 1 and Theorem 2 show that, within our model of computation, the complexity of computing the Delaunay tessellation from an arbitrary triangulation of a (closed, orientable) polyhedral surface is polynomial in the number of triangles and in the logarithm of the aspect ratio of the input triangulation.

1.3 Overview of the proof of Theorem 1

The proof of Theorem 2 is deferred to the full version of the paper. This extended abstract focuses on the proof of Theorem 1, of which we now provide an overview.

[2] Given a set V of points of the surface, finite, non-empty, and containing all the singularities, our results easily extend to Delaunay triangulations whose vertex set is V, but this is incidental to us.

We introduce a slight variation of happiness, more suitable to our needs, which we call *segment-happiness*. To prove Theorem 1, the crux of the matter is to replace the input triangular portalgon by another triangular portalgon of the same surface, whose segment-happiness is "low". For this purpose, our approach is to first focus on portalgons T whose surface $\mathcal{S}(T)$ is *flat*: the interior of $\mathcal{S}(T)$ has no singularity. Note that here we allow $\mathcal{S}(T)$ to have boundary, and this boundary may have singularities. The *systole* of $\mathcal{S}(T)$ is the smallest length of a non-contractible geodesic closed curve in $\mathcal{S}(T)$. Our key technical result is:

▶ **Proposition 3.** *Let T be a portalgon of n triangles, whose sides are all smaller than $L > 0$. Assume that $\mathcal{S}(T)$ is flat. Let $s > 0$ be smaller than the systole of $\mathcal{S}(T)$. One can compute in $O(n \log^2(n) \cdot \log^2(2 + L/s))$ time a portalgon of $O(n \cdot \log(2 + L/s))$ triangles, whose surface is isometric to that of T, and whose segment-happiness is $O(\log(n) \cdot \log^2(2 + L/s))$.*

Sections 3–5 are devoted to the proof of Proposition 3. In Section 3 we focus on particular portalgons, whose surfaces are all homeomorphic to an annulus; the definitions and results of this section are used by the algorithm of Proposition 3. In Section 4 we describe the algorithm for Proposition 3. It is a finely tuned combination of elementary operations such as inserting and deleting edges and vertices in graphs. While the algorithm itself is relatively simple, its analysis is more involved, and is sketched in Section 5. In this section, we first provide a combinatorial analysis, and then we prepare for the geometric analysis by introducing a new parameter on the simple geodesic paths e of a flat surface, *enclosure*, possibly of independent interest. Informally, e is enclosed when a short non-contractible loop can be attached to a point of e not too close to the endpoints of e. We then use enclosure to analyze the algorithm from a geometric point of view, proving Proposition 3.

We then extend Proposition 3 from flat surfaces to surfaces having singularities in their interior, essentially by cutting out caps around these singularities. To get a cleaner result, we also replace $2 + L/s$ by the aspect ratio of T, and we replace segment-happiness by happiness, obtaining:

▶ **Proposition 4.** *Let T be a portalgon of n triangles, of aspect ratio r. One can compute in $O(n \log^2(n) \cdot \log^2(r))$ time a portalgon of $O(n \cdot \log(r))$ triangles, whose surface is $\mathcal{S}(T)$, and whose happiness is $O(n \log(n) \cdot \log^2(r))$.*

The proof of Proposition 4 is omitted from this extended abstract. We have not discussed Delaunay tessellations yet. Still, we are almost ready to prove Theorem 1. Indeed, once we have a portalgon of low happiness, we can compute shortest paths on the surface. And, as already mentioned, shortest path algorithms classically extend to construct Voronoi diagrams and then Delaunay tessellations. Formally:

▶ **Proposition 5.** *Let T be a portalgon of n triangles, of happiness h, such that $\mathcal{S}(T)$ is closed. One can compute the portalgon of the Delaunay tessellation of $\mathcal{S}(T)$ in $O(n^2 h \log(nh))$ time.*

The proof of Proposition 5 is omitted from this extended abstract. We insist that the proof of Proposition 5 is incidental to us, and Proposition 5 is not surprising at all. Our contribution is really the proof of Proposition 4. Theorem 1 is immediate from Proposition 4 and Proposition 5:

Proof of Theorem 1. Proposition 4 computes in $O(n \log^2(n) \cdot \log^2(r))$ time a portalgon T' of $O(n \cdot \log(r))$ triangles, whose happiness is $O(n \log(n) \cdot \log^2(r))$. Proposition 5 then computes the portalgon of the Delaunay tessellation from T' in $O(n^3 \log^2(n) \cdot \log^4(r))$ time. ◀

2 Preliminaries

We use without review standard notions of graph theory and low dimensional topology and geometry, referring to textbooks for details [6, 1, 30, 7]. We only mention that on a surface S, a path $p : [0, 1] \to S$ is **simple** if its restriction to the interval $(0, 1)$ is injective, in which case the image of $(0, 1)$ by p is the **relative interior** of p. We denote by $\ell(p)$ the length of a geodesic path p. Throughout the paper, logarithms are in base two.

The definition of Delaunay tessellation given by Bobenko and Springborn [3, Section 2] is not used in this extended abstract.

2.1 Portalgons, tessellations, and polyhedral surfaces

A **portalgon** T is a disjoint collection of oriented polygons in the Euclidean plane, together with a partial matching of the sides of the polygons such that any two matched sides have the same length. It is **triangular** if all polygons are triangles. See Figure 1. Any subset of the polygons defines a **sub-portalgon** T' of T: two sides of polygons are matched in T' if and only if they are matched in T. In a portalgon T, identifying the matched sides, isometrically, and respecting the orientations of the polygons, provides **the surface of** T, denoted $\mathcal{S}(T)$; it is a 2-dimensional Riemannian manifold whose metric may have singularities. The sides of the polygons of T correspond to a graph T^1 embedded on $\mathcal{S}(T)$, the **1-skeleton** of T.

A **polyhedral surface** is any Riemannian manifold S (possibly with singularities) isometric to the surface of a portalgon. And when we say that a portalgon T is **a portalgon of** S, we implicitly fix an isometry between $\mathcal{S}(T)$ and S. A **tessellation** of S is any 1-skeleton of a portalgon of S, it is a **triangulation** if the portalgon is triangular.

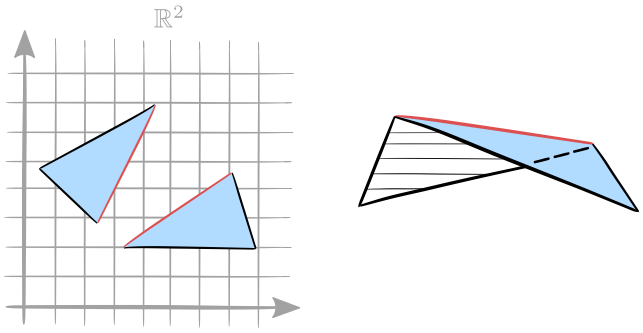


Figure 1 (Left) A triangular portalgon T: two triangles in the Euclidean plane, with two sides matched in red. (Right) The surface $\mathcal{S}(T)$, and the 1-skeleton T^1.

Consider a polyhedral surface S, a triangulation T^1 of S, a vertex x of T^1, and the sum a of the angles of faces of T^1 around x. The point x is a **singularity** if x lies in the boundary of S and $a \neq \pi$, or if x lies in the interior of S and $a \neq 2\pi$. Every other point of S is **flat**. This does not depend on any particular triangulation of S. A surface S is **flat** if its interior has no singularity (although its boundary may have singularities). The closed flat surfaces are called **flat tori**.

2.2 Aspect ratio, systole, happiness, and segment-happiness

The **aspect ratio** of a triangular portalgon T is the maximum side length of a triangle of T divided by the smallest height of a triangle of T (possibly another triangle). Note that the aspect ratio is always greater than or equal to $2/\sqrt{3} > 1$, because the maximum side length of a triangle is always greater than or equal to $2/\sqrt{3}$ times its smallest height.

The **systole** of a polyhedral surface S is the smallest length of a non-contractible geodesic closed curve in S, except in the particular case where every closed curve in S is contractible, in which case the systole is ∞. The important thing is that for every positive real s smaller than the systole of S, any non-contractible closed curve in S is longer than s.

The **happiness** of a portalgon T is the maximum number of times a shortest path in $\mathcal{S}(T)$ visits the image of a polygon of T, maximized over all the shortest paths of $\mathcal{S}(T)$ and all the polygons of T (see [20, Section 3]). We introduce a variation, more suitable to our needs. In a polyhedral surface S, a **segment** is a simple geodesic path e whose relative interior is disjoint from any singularity of S. The **segment-happiness** of e in S, denoted $h_S(e)$, is the maximum number of intersections between e and a shortest path of S, maximized over all the shortest paths of S. The **segment-happiness** of a portalgon T is then the maximum segment-happiness $h_{\mathcal{S}(T)}(e)$, maximized over the edges e of its 1-skeleton T^1. A priori, the segment-happiness of a portalgon T differs from the happiness of T. Indeed a path in $\mathcal{S}(T)$ may visit many times a face of T^1 without intersecting any edge of T^1 more than once, if the face has high degree. However, if T is triangular, then the happiness and the segment-happiness of T do not differ by more than a constant factor.

3 Tubes and bifaces

In this section we focus on particular triangular portalgons. See Figure 2. A **tube** is a triangular portalgon X whose surface $\mathcal{S}(X)$ is homeomorphic to an annulus and has no singularity in its interior, and whose 1-skeleton X^1 has exactly one vertex on each boundary component of $\mathcal{S}(X)$. Among tubes, a **biface** is a portalgon B of two triangles whose respective sides s_0, s_1, s_2 and s'_0, s'_1, s'_2, in order (clockwise say), are such that s_0 is matched with s'_0 and s_1 is matched with s'_1. Its 1-skeleton B^1 has four edges: two loop edges forming the two boundary components of $\mathcal{S}(B)$, which we call **boundary edges**, and two edges whose relative interiors are included in the interior of $\mathcal{S}(B)$, which we call **interior edges**.

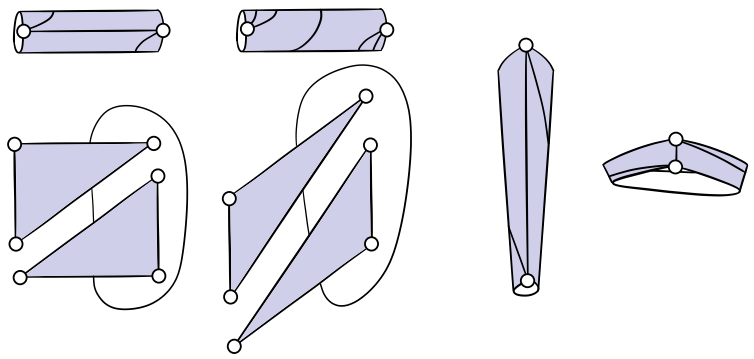


Figure 2 (From left to right) A good biface, a biface not good, a thin biface, a thick biface.

We say that a biface B is **good** if the two interior edges e and f of B^1 satisfy both of the following up to possibly exchanging e and f. First, e is a shortest path in $\mathcal{S}(B)$. Second, cut $\mathcal{S}(B)$ along e, and consider the resulting quadrilateral. If this quadrilateral has two diagonals then f is shortest among the two diagonals. We will distinguish good bifaces. A good biface B is **thin** if every interior edge of B^1 is longer than every boundary edge of B^1. Otherwise B is **thick**. While tubes and bifaces have unbounded happiness, good bifaces on the other hand satisfy the following:

▶ **Lemma 6.** *Given a good biface B, let e be an interior edge of B^1. Then $h_{\mathcal{S}(B)}(e) \leq 6$.*

We will use the elementary operation of replacing a tube by a good biface:

▶ **Proposition 7.** *Let X be a tube with n triangles, whose sides are smaller than $L > 0$. Let $s > 0$ be smaller than the systole of $\mathcal{S}(X)$. One can compute a good biface whose surface is $\mathcal{S}(X)$ in $O(n \log n \cdot \log(2 + L/s))$ time.*

Proposition 7 is similar to a result described by Löffler, Ophelders, Silveira, and Staals [20, Theorem 45], building upon a ray shooting algorithm of Erickson and Nayyeri [9].

4 Description of the algorithm

In this section we describe our algorithm for Proposition 3. We first describe the elementary operations and the data structure, before giving the algorithm itself. Along the way, we provide informal explanations of our choices. We do not prove anything, as the analysis of the algorithm is deferred to Section 5.

4.1 Inserting vertices and edges

Informally, our goal is to "improve the geometry" of a triangular portalgon T. We will make this precise in Section 5. Roughly, the issue is that, without any condition on T, the edges of T^1 that lie in the interior of $\mathcal{S}(T)$ can be arbitrarily long, so one of them may intersect some shortest path arbitrarily many times by wrapping around the surface, and so the segment-happiness of T can be arbitrarily large. A naive way of shortening an edge is to cut the edge in two at its middle point.

InsertVertices. Given a triangular portalgon T, consider every edge e of T^1 that lies in the interior of $\mathcal{S}(T)$, and insert the middle point of e as a vertex in T^1.

Applying **InsertVertices** to a *triangular* portalgon T produces a portalgon T' whose polygons are usually not triangles. We now consider transforming T' into a triangular portalgon. To do that we repeatedly cut the polygons of T'. We need a definition. In the plane consider a polygon P, two vertices $u \neq v$ of P, and the rectilinear segment a between u and v. If the relative interior of a is included in the interior of P then a is called a vertex-to-vertex arc of P. It is easily seen that if P is not a triangle then P has at least one vertex-to-vertex arc. Among the vertex-to-vertex arcs of P, the shortest ones are the **shortcuts** of P. We emphasize that we consider the shortest ones among all the vertex-to-vertex arcs, without fixing the endpoints, but the endpoints are chosen among the vertices of P. In a portalgon T every polygon P corresponds to a face F of T^1, and every shortcut of P corresponds to a path whose relative interior is included in F: we say of this path that it is a shortcut of F.

InsertEdges. Given a portalgon T, as long as there is a face of T^1 that is not a triangle, insert a shortcut of this face as an edge in T^1.

We shall apply **InsertVertices** followed by **InsertEdges** to a triangular portalgon T in order to produce another triangular portalgon T', hopefully with a "nicer geometry". The problem is now that T'^1 has more vertices than T^1. All the other operations of the algorithm are devoted to keeping the number of vertices low.

4.2 Deleting vertices

From now on it is important that every surface considered is flat, there is no singularity in its interior. Given a triangular portalgon T, assuming that the surface $\mathcal{S}(T)$ is flat, we consider decreasing the number of vertices of T^1. To do that we naturally consider deleting some vertices. Not all vertices can be deleted. For example a vertex incident to a loop edge cannot be deleted. Also we will not delete vertices that lie on the boundary of the surface $\mathcal{S}(T)$. A vertex of T^1 is **weak** if it lies in the interior of $\mathcal{S}(T)$ and is not incident to any loop edge in T^1. It is **strong** otherwise.

DeleteVertices. Given a triangular portalgon T whose surface $\mathcal{S}(T)$ is flat, construct a maximal independent set V of weak vertices of T^1 that have degree smaller than or equal to six. For every vertex $v \in V$ delete v and its incident edges from T^1.

After performing **DeleteVertices**, the polygons of T are usually not triangles anymore, but this will be solved by applying **InsertEdges** after each application of **DeleteVertices**. Observe that in **DeleteVertices** we delete only vertices of degree smaller than or equal to six. Informally, the reason is that deleting a weak vertex of degree $d \geq 3$ creates a face of degree d around it. We then insert $d-3$ edges in this face when applying **InsertEdges**. The problem is that only a constant number of edges can be inserted in each face without risking to destroy our improvements on the geometry of the tessellation. This is why we make sure that $d = O(1)$ beforehand. The exact bound on d is not really important (although changing it would change some constants of the algorithm), but it must be at least six so that we can still remove a fraction of the excess vertices this way, at least when most of them are strong. Similar ideas can be found in the literature, see for example Kirkpatrick [16, Lemma 3.2].

4.3 Simplifying tubes

The operation **DeleteVertices** cannot delete strong vertices, and among them the vertices that lie the interior of the surface and are incident to a loop edge. In this section we describe an operation for deleting such vertices.

In order to grasp the intuition, observe, informally, that it is possible that almost all the vertices of T^1 lie in the interior of $\mathcal{S}(T)$ and are incident to a loop edge. Fortunately, it turns out that in this case there must be a sub-portalgon X of T such that X is a tube and the interior of $\mathcal{S}(X)$ contains loop edges of X^1. We delete such loop edges by replacing X by a good biface with Proposition 7. There is one subtlety: we must choose X carefully so that we replace any concatenation of tubes by a single biface when possible, in order to delete the loops in-between the tubes, instead of replacing each tube individually. That leads to:

SimplifyTubes. In a triangular portalgon T whose surface $\mathcal{S}(T)$ is flat, do the following:

1. In T^1 build a set J of loop edges that lie in the interior of $\mathcal{S}(T)$ and are pairwise disjoint, as follows. There are two cases:
   - **a.** If $\mathcal{S}(T)$ is homeomorphic to a torus, do the following. Let J contain two disjoint loop edges of T^1 if there exist two such edges, otherwise let $J = \emptyset$.
   - **b.** Otherwise do the following. Construct a set J' of loop edges by considering every vertex v of T^1 that lies in the interior of $\mathcal{S}(T)$ and is incident to a loop edge, and by putting one of the loop edges incident to v in J'. Then build a subset $J \subseteq J'$ by removing from J' every $e \in J'$ satisfying both of the following. First, cutting $\mathcal{S}(T)$

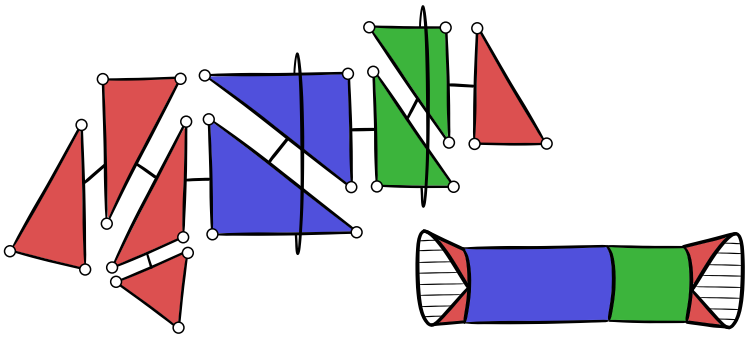


Figure 3 Data structure for **Algorithm**: a portalgon whose polygons are partitioned, here by color, inducing sub-portalgons called regions, and a region singularized as active, here in red.

along the loops in J', and considering the resulting connected components, two such components are adjacent to e (instead of one), say S_0 and S_1. Second, each one of the two sub-portalgons of T whose surfaces are S_0 and S_1 is a tube.

2. Cut the surface $\mathcal{S}(T)$ along the loops in J. Each resulting component is the surface of a sub-portalgon X of T. If X is a tube replace X by a good biface B.

The idea behind step 1b is to remove loops from J' so that step 2 replaces a concatenation of tubes by a single good biface when possible, instead of replacing the tubes individually.

4.4 Data structure for marking bifaces as inactive

We are almost ready to give the algorithm, but there is still one important thing to describe. In step 2 of **SimplifyTubes**, if the good biface B is *thin* we will not just replace X by B, but we will also make sure to not modify B ever again. In this sense B becomes inactive. Doing so requires a data structure remembering which parts of the portalgon are inactive.

See Figure 3. The data structure maintains a portalgon R together with a partition of the polygons of R. Each set X of polygons in the partition defines a sub-portalgon of R which we call **region**. One region is singularized as the **active** region R_A. The other regions are **inactive**. Note that the surface of the active region may be disconnected, and that the surfaces of distinct inactive regions may be adjacent.

The data structure will be initialized by setting $R_A = R$, without inactive region. Then the algorithm will apply the routines **InsertVertices**, **InsertEdges**, **DeleteVertices**, and **SimplifyTubes** to the active region R_A, and mark as inactive every thin biface encountered in step 2 of **SimplifyTubes**. The surface of R_A will diminish over time as more and more regions are marked inactive. This may increase the numbers of connected components and boundary components of $\mathcal{S}(R_A)$, ruining our efforts to keep the combinatorial complexity of R_A bounded. To counteract this, we introduce:

Gardening. Every connected component of $\mathcal{S}(R_A)$ is the surface of a sub-portalgon X of R_A. If X is a tube replace X by a good biface B, and mark B as inactive.

We described everything that the algorithm can do to the data structure. This immediately implies three invariants maintained by the algorithm. First, 1) Every polygon of the active region has degree at most six, and 2) Every inactive region is a good biface. For the last invariant we need a definition. Recall that in R if two sides s and s' of polygons are matched

then s and s' correspond to an edge e of R^1. If moreover s and s' belong to different polygons, and if their respective polygons belong to different regions, we say that e is **separating**. Then e is a loop, for it is a boundary edge of a biface by 2), and e belongs to the interior of $\mathcal{S}(R)$. The third invariant is that 3) The separating loops are pairwise disjoint (no two of them are based at the same vertex of R^1).

4.5 Algorithm

The algorithm repeatedly applies two parts. The first part "improves the geometry" by applying **InsertVertices** and then **InsertEdges**. However this increases the number of vertices. So the second part applies **SimplifyTubes**, **DeleteVertices**, and **InsertEdges**, together with **Gardening**. The second part can only remove a fraction of the vertices at once, so it is repeated several times. It turns out that 350 repetitions suffice.

Algorithm. Given a triangular portalgon T whose surface $\mathcal{S}(T)$ is flat, and $N \geq 1$, do the following. Initialize the data structure by letting R be the input portalgon T, and by letting the active region R_A be R itself, without inactive region. Repeat N times the following:

1. Apply **InsertVertices** to R_A. Then apply **InsertEdges** to R_A.
2. Repeat 350 times the following:
   a. Apply **Gardening**. Then apply **SimplifyTubes** to R_A but in step 2 of **SimplifyTubes**, whenever B is thin, mark B as inactive. Apply **Gardening** again.
   b. Apply **DeleteVertices** to R_A. Then apply **InsertEdges** to R_A.

In the end return R.

When proving Proposition 3, we will apply **Algorithm** with $N = \lceil \log(2 + L/s) \rceil$.

5 Analysis of the algorithm

In this section, we sketch the analysis of **Algorithm** (Section 4) to prove Proposition 3.

5.1 Combinatorial analysis

▶ **Proposition 8.** *Apply* **Algorithm** *to a portalgon T of n triangles, whose surface $\mathcal{S}(T)$ is flat. During the execution the number of polygons of the active region R_A is $O(n)$.*

In this extended abstract, we only sketch the proof of Proposition 8.

Sketch of proof. We consider R^1_A, the 1-skeleton of the active region R_A, and we show that the number m_A of vertices of R^1_A remains $O(n)$ throughout the execution. There are two loops in the algorithm: the main loop, which repeats N times, and the interior loop, which repeats 350 times within each iteration of the main loop. To prove the lemma, we consider a single iteration of the main loop, we assume that m_A exceeds n by at least a constant factor at the beginning of the iteration, and we prove that m_A has decreased after the iteration.

The iteration starts with **InsertVertices**. This is the only moment where m_A may increase, and we prove that m_A is multiplied by at most a constant factor. Then the iteration applies the interior loop, and we claim that, as long as m_A exceeds n by a constant factor, m_A is divided by at least a constant factor by each iteration of the interior loop. We show that this claim implies the lemma as the interior loop is applied sufficiently many times to counteract the initial increase of m_A. To prove the claim, we show that for **DeleteVertices** to remove a fraction of the vertices of R^1_A, it suffices that m_A vastly exceeds the genus and the number of boundary components of $\mathcal{S}(R_A)$, and that almost all of the vertices of R^1_A are weak. We show that this is ensured by first applying **Gardening** and **SimplifyTubes**. ◀

5.2 Enclosure

To analyze **Algorithm** from a geometric point of view, we introduce, on the segments of a flat surface S, a parameter that we call *enclosure*. So consider a segment e of S. See Figure 4.

Informally, e is "enclosed" in S when a short non-contractible loop can be attached to a point of e not too close to the endpoints of e. Formally, consider a point x in the relative interior of e. We denote by $\langle x \rangle_e$ the minimum length of the two sub-segments of e separated by x. Assume that there exists a loop γ based at x in S, such that γ is geodesic except possibly at its basepoint. Further assume that its length satisfies $\ell(e) < \langle x \rangle_e$. In this case γ and e are necessarily in *general position*: informally, they do not overlap, more formally, every sufficiently short sub-path of γ is either disjoint from e or its intersection with e is a single point. There are two cases: either γ crosses e at x, or γ meets x on only one side of e. If γ crosses e at x, then we say that γ **encloses** e **in** S. Also we say that γ encloses e **by a factor of** $\langle x \rangle_e / \ell(\gamma)$ in S. The **enclosure** $c_S(e) \geq 1$ is the supremum of the ratios $\langle x \rangle_e / \ell(\gamma)$ over all the basepoints x in the relative interior of e, and over all the loops γ based at x that enclose e in S. It is conventionally set to one if there is no loop enclosing e in S.

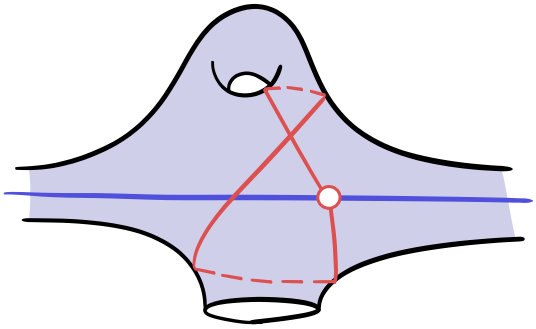


Figure 4 The red loop encloses the blue segment in the surface.

The segment-happiness $h_S(e)$ and the length $\ell(e)$ can be bounded from above using the enclosure $c_S(e)$. Our bound depends on the surface S. More precisely, on the systole of S and the diameter of S. But instead of the diameter of S, we consider a triangulation of S, and we use its number n of triangles together with the maximum length L of its edges. This will be more convenient to us when analyzing **Algorithm**. We prove:

▶ **Proposition 9.** *Let e be a segment of S. Let $s > 0$ be smaller than the systole of S. Assume that there is a triangulation of S with $n \geq 1$ triangles, whose edges are all smaller than $L > 0$. Then $h_S(e) = O(c_S(e) \cdot (1 + \log c_S(e) + \log n + \log\lceil L/s \rceil))$ and $\ell(e)/s = O(c_S(e) \cdot n \cdot \lceil L/s \rceil^2)$.*

In Proposition 9 the $O()$ notation does not depend on S, it involves a universal constant. In the second inequality of Proposition 9 the exact powers above $\lceil L/s \rceil$ and n, here 2 and 1, do not matter to us. We need only a polynomial in $\lceil L/s \rceil$ and n.

5.3 Geometric analysis

The geometric analysis of **Algorithm** consists in two properties on the enclosure and the length of the edges involved in any execution: Lemma 10 and Proposition 12 below, whose proofs we sketch in Section 5.3.1 and Section 5.3.2. Each proof relies on properties of enclosure that are independent of **Algorithm**, or of any portalgon, and can be seen as independent mathematical contributions of us. In this extended abstract we only explain how these properties of enclosure serve to analyse **Algorithm**.

In this section, we fix a portalgon T of n triangles, whose sides are smaller than some positive real L, and whose surface $\mathcal{S}(T)$ is flat. We abbreviate $S = \mathcal{S}(T)$. We apply the algorithm **Algorithm** to T, and we discuss the execution of the algorithm.

5.3.1 The separating loops are not very enclosed

▶ **Lemma 10.** *Any time during the execution every separating loop e satisfies $c_S(e) \leq 2$.*

Lemma 10 follows from the following property of enclosure:

▶ **Proposition 11.** *Assume that S contains the surface of a thin biface B, and let e be one of the two boundary edges of B^1. Then $c_S(e) \leq 2$.*

Proof of Lemma 10. Only step 2 of **SimplifyTubes** may create a separating loop, by marking a *thin* biface B as inactive. Then B is is never touched again by the algorithm. So the algorithm maintains the invariant that every separating loop e is adjacent to the surface of at least one inactive region that is a *thin* biface. So $c_S(e) \leq 2$ by Proposition 11. ◀

5.3.2 The very enclosed edges shorten exponentially fast

▶ **Proposition 12.** *After $i \geq 1$ iterations of the main loop, let e be an edge of R^1_A. If $c_S(e) > 22000 \cdot i$ then $\ell(e) < 2^{1-i}L$.*

To prove Proposition 12, we analyze each routine applied. Informally, each application of **InsertVertices** "improves the geometry" of the active region, and the rest of the algorithm does not deteriorate this improvement too much. Formally:

▶ **Lemma 13.** *Consider the active regions R_A and R'_A respectively before and after some application of* **InsertVertices**. *Assume that there is an edge e' of R'^1_A such that $c_S(e') > 2$. Then there is an edge e of R^1_A such that $c_S(e) \geq c_S(e')$ and $\ell(e) \geq 2\ell(e')$.*

Lemma 13 follows from the following (easy) property of enclosure:

▶ **Lemma 14.** *Let $f \subseteq e$ be segments in S. Then $c_S(e) \geq c_S(f)$.*

Proof of Lemma 13. First observe that e' is not included in the boundary of $\mathcal{S}(R'_A)$ because e' is enclosed and thus not included in the boundary of S, and because e' is not a separating loop by Lemma 10. So there is an edge e of R^1_A such that e' is one of the two half-segments obtained after the insertion of the middle point of e as a vertex. Then $\ell(e) = 2\ell(e')$. And $c_S(e) \geq c_S(e')$ by Lemma 14. ◀

▶ **Lemma 15.** *Consider the active regions R_A and R'_A respectively before and after some application of* **InsertEdges**. *Assume that there is an edge e' of R'^1_A such that $c_S(e') > 14$. Then there is an edge e of R^1_A such that $c_S(e) \geq c_S(e') - 12$ and $\ell(e) \geq (1 - 12/c_S(e')) \cdot \ell(e')$.*

Lemma 15 follows from the following (key) property of enclosure:

▶ **Proposition 16.** *Let F be a face of a tessellation of S. Assume that F has a shortcut e such that $c_S(e) > 6$. Then F has a side f such that $c_S(f) \geq c_S(e) - 4$ and $\ell(f) \geq (1 - 4/c_S(e)) \cdot \ell(e)$.*

Proof of Lemma 15. Here we crucially use the fact that every polygon of R_A has degree at most six. so that at most three edges are inserted within the polygon. Indeed either e' was already an edge of R^1_A and there is nothing to do, or e' has been inserted in some face F of R^1_A. At most three edges were inserted in F, and Proposition 16 applied at most three times gives a boundary edge e of F such that $c_S(e) \geq c_S(e') - 12$ and $\ell(e) \geq (1 - 12/c_S(e'))\ell(e')$. ◀

▶ **Lemma 17.** *Consider the active regions R_A and R'_A respectively before and after some application of* **SimplifyTubes**. *Assume that there is an edge e' of R'^1_A such that $c_S(e') > 6$. Then there is an edge e of R^1_A such that $c_S(e) \geq c_S(e') - 5$ and $\ell(e) \geq (1 - 4/c_S(e')) \cdot \ell(e')$.*

Lemma 17 is similar to Lemma 15, its proof is omitted in this extended abstract.

Proof of Proposition 12. Consider the active regions R_A and R'_A respectively at the beginning of the algorithm, and after i iterations of the main loop. Assume that there is an edge e' in R'^1_A such that $c_S(e') > 22000 \cdot i$. During those i iterations there has been i applications of **InsertVertices**, $351i$ applications of **InsertEdges**, and $350i$ applications of **SimplifyTubes**. Also $12 \cdot 351i + 5 \cdot 350i < 11000i$. So Lemma 13, Lemma 15, and Lemma 17 imply that there is an edge e in R^1_A such that $\ell(e) \geq 2^i(1 - 11000i/c_S(e'))\ell(e') > 2^{i-1}\ell(e')$. And $\ell(e) \leq L$ because e belongs to the input triangulation T^1. ◀

5.4 Proof of Proposition 3

We need a last (easy) lemma:

▶ **Lemma 18.** *Let S be a flat surface. Assume that S contains the surface of a tube X. Then the systole of $\mathcal{S}(X)$ is greater than or equal to the systole of S.*

Proof of Proposition 3. Apply **Algorithm** to T with $N = \lceil \log(2 + L/s) \rceil$, resulting in a triangular portalgon R. By Proposition 8 the number of polygons of the active region is $O(n)$ throughout the execution. So in the end R has $O(n \cdot \log(2 + L/s))$ triangles; Indeed each iteration of the main loop marks $O(n)$ triangles as inactive, and there are $\lceil \log(2 + L/s) \rceil$ iterations of the main loop. We have two claims that immediately imply the proposition.

Our first claim is that the algorithm takes $O(n \log^2(n) \cdot \log^2(2 + L/s))$ time. Let us prove this first claim. Each application of **InsertVertices** or **InsertEdges** takes $O(n)$ time. And each application of **SimplifyTubes** or **Gardening** takes $O(n \log(n) \cdot \log(2 + \Lambda/s))$ time by Proposition 7 and Lemma 18, where Λ is the maximum length reached by an edge of the 1-skeleton of the active region during the execution. Now let us bound Λ. If at some point an edge e of the 1-skeleton of the active region is longer than L then $c_S(e) = O(\log(2 + L/s))$ by Proposition 12. Moreover $\ell(e)/s = O(c_S(e) \cdot n\lceil L/s \rceil^2)$ by Proposition 9. This proves $\log(2 + \Lambda/s) = O(\log(n) + \log(2 + L/s))$, which proves the claim.

Our second claim is that in the end every edge e of R^1 satisfies $h_S(e) = O(\log(n) \cdot \log^2(2 + L/s))$. Let us prove this second claim. First observe that if e is in R^1_A then $c_S(e) < 22000 \log(2 + L/s)$, for otherwise Proposition 12 would imply $\ell(e) < 2s$, implying that no loop encloses e in S, a contradiction. In this case $h_S(e) = O(\log(2 + L/s) \cdot (\log(n) + \log(2 + L/s)))$ by Proposition 9, and we are done. Every other edge of R^1 belongs to the 1-skeleton of an inactive good biface B. Every boundary edge e of B^1 is either a boundary component of S or a separating loop, so $c_S(e) \leq 2$ by Lemma 10, and so $h_S(e) = O(\log(n) + \log(2 + L/s))$ by Proposition 9. Every interior edge f of B^1 then satisfies $h_S(f) = O(\log(n) + \log(2 + L/s))$ by Lemma 6. This proves the second claim, and the proposition. ◀

References

1 Mark Anthony Armstrong. *Basic Topology*. Springer Science & Business Media, 2013.

2 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

3 Alexander I. Bobenko and Boris A. Springborn. A discrete Laplace–Beltrami operator for simplicial surfaces. *Discrete & Computational Geometry*, 38(4):740–756, 2007. doi:10.1007/S00454-007-9006-1.

4 Jindong Chen and Yijie Han. Shortest paths on a polyhedron, part I: Computing shortest paths. *International Journal of Computational Geometry & Applications*, 6(02):127–144, 1996.

5 Mark De Berg. *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

6 Reinhard Diestel. *Graph Theory*. Springer-Verlag, 2000.

7 Manfredo Perdigao Do Carmo. *Riemannian Geometry*, volume 2. Springer, 1992.

8 Jeff Erickson. Ernie's 3D pancakes: Shortest paths on PL surfaces. `https://3dpancakes.typepad.com/ernie/2006/03/shortest_paths_.html`, 2006.

9 Jeff Erickson and Amir Nayyeri. Tracing compressed curves in triangulated surfaces. In *Proceedings of the twenty-eighth annual symposium on Computational geometry*, pages 131–140, 2012. `doi:10.1145/2261250.2261270`.

10 Jeff Erickson, Ivor Van Der Hoog, and Tillmann Miltzow. Smoothing the gap between NP and ER. *SIAM Journal on Computing*, pages FOCS20–102, 2022.

11 Steven Fortune. Voronoi diagrams and delaunay triangulations. In *Handbook of discrete and computational geometry*, pages 705–721. Chapman and Hall/CRC, 2017.

12 Eugene Gutkin and Chris Judge. Affine mappings of translation surfaces: geometry and arithmetic. *Duke Mathematical Journal*, 2000.

13 Paul S Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. Siggraph, 1997.

14 Pascal Hubert and Thomas A Schmidt. An introduction to veech surfaces. *Handbook of dynamical systems*, 1(200601), 2006.

15 Claude Indermitte, Th M Liebling, Marc Troyanov, and Heinz Clémençon. Voronoi diagrams on piecewise flat surfaces and an application to biological growth. *Theoretical Computer Science*, 263(1-2):263–274, 2001. `doi:10.1016/S0304-3975(00)00248-6`.

16 David Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983. `doi:10.1137/0212002`.

17 Hsueh-Ti Derek Liu, Mark Gillespie, Benjamin Chislett, Nicholas Sharp, Alec Jacobson, and Keenan Crane. Surface simplification using intrinsic error metrics. *ACM Transactions on Graphics*, 42(4), 2023. `doi:10.1145/3592403`.

18 Yong-Jin Liu, Zhanqing Chen, and Kai Tang. Construction of iso-contours, bisectors, and voronoi diagrams on triangulated surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1502–1517, 2010. `doi:10.1109/TPAMI.2010.221`.

19 Yong-Jin Liu, Chun-Xu Xu, Dian Fan, and Ying He. Efficient construction and simplification of delaunay meshes. *ACM Transactions on Graphics (TOG)*, 34(6):1–13, 2015. `doi:10.1145/2816795.2818076`.

20 Maarten Löffler, Tim Ophelders, Rodrigo I. Silveira, and Frank Staals. Shortest paths in portalgons. In *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258, pages 48:1–48:16, 2023. `doi:10.4230/LIPIcs.SoCG.2023.48`.

21 Howard Masur. Ergodic theory of translation surfaces. *Handbook of dynamical systems*, 1:527–547, 2006.

22 Joseph S.B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987. `doi:10.1137/0216045`.

23 David M. Mount. *Voronoi Diagrams on the Surface of a Polyhedron*. University of Maryland, 1985.

24 Jim Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of algorithms*, 18(3):548–585, 1995. `doi:10.1006/JAGM.1995.1021`.

25 Nicholas Sharp and Keenan Crane. You can find geodesic paths in triangle meshes by just flipping edges. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. `doi:10.1145/3414685.3417839`.

26 Nicholas Sharp, Mark Gillespie, and Keenan Crane. Geometry processing with intrinsic triangulations. *SIGGRAPH'21: ACM SIGGRAPH 2021 Courses*, 2021.

27 Nicholas Sharp, Yousuf Soliman, and Keenan Crane. Navigating intrinsic triangulations. *ACM Transactions on Graphics (TOG)*, 38(4):1–16, 2019. doi:10.1145/3306346.3322979.

28 Jonathan Richard Shewchuk. *Delaunay refinement mesh generation*. Carnegie Mellon University, 1997.

29 Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational geometry*, 22(1-3):21–74, 2002. doi:10.1016/S0925-7721(01)00047-5.

30 John Stillwell. *Classical topology and combinatorial group theory*. Springer-Verlag, New York, second edition, 1993.

31 Kenshi Takayama. Compatible intrinsic triangulations. *ACM Transactions on Graphics (TOG)*, 41(4):1–12, 2022. doi:10.1145/3528223.3530175.

32 Anton Zorich. Flat surfaces. *arXiv preprint*, 2006. arXiv:math/0609392.

The Depth Poset Under Transpositions in the Filter

Herbert Edelsbrunner
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Michał Lipiński
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Marian Mrozek
Division of Computational Mathematics, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Manuel Soriano-Trigueros
Universidad de Sevilla, Spain
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Fedor Zimin
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Abstract

The depth poset of a filtered Lefschetz complex reflects the dependencies between the cancellations of different shallow birth-death pairs. Using the fast algorithms for computing the depth poset in [9] and for updating the persistence diagram under transpositions in [7], we give a complete case analysis of how transpositions of cells in the filter affect the depth poset. In addition, we present statistics on the depth poset for random point data and its sensitivity to the transpositions that occur in random straight-line homotopies.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Algebraic topology, Lefschetz complexes, persistent homology, vines and vineyards, birth-death pairs, shallow pairs, relations, partial orders, transpositions

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.41

Related Version *Previous Version*: https://arxiv.org/abs/2511.21961

Funding *Herbert Edelsbrunner*: DFG Collaborative Research Center TRR 109, Austrian Science Fund (FWF), grant no. I 02979-N35
Michał Lipiński: European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 101034413
Marian Mrozek: Polish National Science Center under Opus Grant 2019/35/B/ST1/00874 and Opus Grant 2025/57/B/ST1/00550

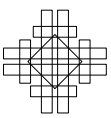
Acknowledgements The authors thank Jakub Leśkiewicz and Bartosz Furmanek for discussions that helped improve the paper.

1 Introduction

The field of *topology optimization* aims at the design of shapes whose large- and small-scale connectivity is subject to modifications applied in order to satisfy constraints and optimize objective functions. The traditional approach is numerical and optimizes through repeated local improvement; see the text by Bendsøe [4] and the major revision by Bendsøe and Sigmund [5]. More recently, the methods of topological data analysis have matured and provide an alternative approach to this seasoned problem; see e.g. the "big steps" algorithm by Nigmetov and Morozov [19], which is closely related to the work presented in this paper. Specifically, we aim at exploring the space of shapes that relate to each other through the



© Herbert Edelsbrunner, Michał Lipiński, Marian Mrozek, Manuel Soriano-Trigueros, and Fedor Zimin;
licensed under Creative Commons License CC-BY 4.0



cancellation and interchange of the critical points of a function. This is naturally related to the simplification of functions and shapes, as studied in [1, 3, 11], but goes beyond this work by casting light on the space of possible simplifications.

An overarching goal for the presented work is to uncover the potential new tools from combinatorial dynamics pioneered by Forman [13, 12], offering discrete counterparts for classic concepts such as the Conley index, Morse decomposition, and connection matrix [16, 17, 18].

The primary technical prerequisites for our work are the depth poset of a filter on a complex, as introduced in [9], the well established concept of persistent homology; see the text by Edelsbrunner and Harer [10], and the vineyard algorithm for computing persistence along a parametrized family of functions, as originally described in [7]. The depth poset collects all dependencies between cancellations of *shallow birth-death pairs* (formerly known as *apparent pairs* [2] and *close pairs* [8]) and thus provides a global yet discrete representation of the space of shapes reachable by sequences of such cancellations. We mention that shallow pairs appeared in other seemingly unrelated contexts, for example in fast adaptive sorting [20]. Our main contribution is the complete case analysis of how transpositions in the filter affect the depth poset. Working toward establishing the depth poset as a first class object, we also present statistics on the size and shape of this poset and its variation under transpositions.

Outline. Section 2 explains the technical background needed for our results. Section 3 gives the complete case analysis of how a transposition in the filter affects the depth poset of the same. Section 4 presents statistical results that probe the depth poset and homotopies between them for random PL height functions on Delaunay mosaics of Poisson point processes.

2 Background

This section presents the background we need to study the relations and orders in which our results are couched. Besides standard material, we need concepts from persistent homology and its matrix reduction algorithms. We refer to [10] for more comprehensive background.

Relations. We adapt the terminology from graph theory, in which a relation consists of *(directed) arcs* that connect *nodes*. Each arc goes from its *source* to its *target*, both of which are nodes. A *(directed) path* is a sequence of arcs such that the target of each arc is the source of the next arc, and it goes from the source of the first arc to the target of the last arc. It is a *directed cycle* if the target of the last arc is also the source of the first arc. The relation is *acyclic* if it has no directed cycle. Note however that an acyclic relation may have *undirected cycles*, such as two disjoint paths that connect the same two nodes.

An acyclic relation has a unique *transitive closure*, which is the possibly larger (acyclic) relation that has an arc from a source to a target whenever there is such a path. A *partial order* is an acyclic relation that is its own transitive closure, and in situations where we wish to emphasize the nodes rather than the arcs, we refer to it as a *partially ordered set*, or a *poset*, for short. Two nodes in a poset are *comparable* if there is an arc that connects them, and we write $a \leq b$ if a is the source and b is the target of the connecting arc. A *total order* is a partial order in which any two nodes are comparable. It is a *linear extension* of a partial order on the same nodes if $a \leq b$ in the partial order implies $a \leq b$ in the total order.

In this paper we work with Lefschetz complexes with coefficients in $\mathbb{Z}/2\mathbb{Z}$ for simplicity, which can be viewed as chain complexes with a fixed bases. In particular, they avoid the complications that come with concrete geometric descriptions of the cells. This type of complex was introduced first under the more modest name of a *complex* in the book by Lefschetz [15]. We borrow the below definition from [9].

▶ **Definition 2.1.** *A* Lefschetz complex *with coefficients in* $\mathbb{Z}/2\mathbb{Z}$ *is a triplet* $(K, \dim, \kappa)$*, in which* K *is a finite set of elements, called* cells*,* $\dim\colon K \to \mathbb{Z}$ *maps each cell to its* dimension*, and* $\kappa\colon K \times K \to \mathbb{Z}/2\mathbb{Z}$ *is the* incidence map *satisfying* $\sum_{y \in K} \kappa(z,y) \cdot \kappa(y,x) = 0$ *for all* $z, x \in K$ *and* $\kappa(y,x) \neq 0$ *only if* $\dim y = \dim x + 1$*. For simplicity, we write just* K *for a Lefschetz complex assuming that* dim *and* κ *are implied. If* $\kappa(y,x) \neq 0$*, then we say that cell* x *is a* facet *of* y *and cell* y *is a* cofacet *of* x*, we denote this relation by* $x \preceq y$*. The transitive closure of the facet relation forms a partial order on* K *called the* face relation*.*

An injective function on a complex, $f\colon K \to \mathbb{R}$, that satisfies $f(a) < f(b)$ whenever $a \prec b$ is called a *filter* of K. A total order in which a precedes b iff $f(a) < f(b)$ is then a linear extension of the face relation; see the left panel of Figure 1.

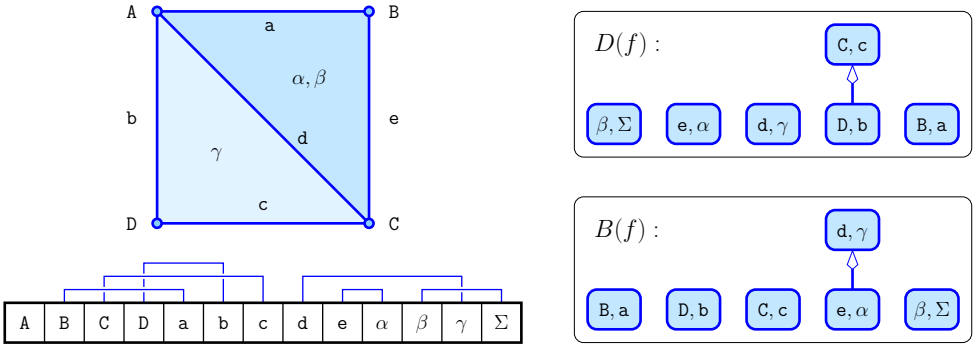


Figure 1 *Upper left:* a complex with four vertices, five edges, three triangles (α, β, γ, in which α and β have the same boundary), and one 3-cell (Σ, which is sandwiched between α and β). *Lower left:* the ordering implied by a filter, with birth-death pairs as indicated (see definition below). *Right:* the death and birth relations on the birth-death pairs as computed by Algorithms 1 and 2 below.

Persistent Homology and Transpositions. Given a filter on a complex, $f\colon K \to \mathbb{R}$, we construct the complex by adding one cell at a time following the implied order. After adding i cells, we arrive at a subcomplex $K_i \subseteq K$, and we refer to the resulting sequence of subcomplexes as a *filtration* of K. Fixing a dimension, p, and a coefficient field, e.g. $\mathbb{Z}/2\mathbb{Z}$, we apply the degree-p homology functor to each subcomplex and thus get a linear sequence of vector spaces connected by linear maps:

$$\mathsf{H}_p(\emptyset) = \mathsf{H}_p(K_0) \to \mathsf{H}_p(K_1) \to \ldots \to \mathsf{H}_p(K_n) = \mathsf{H}_p(K); \tag{1}$$

see e.g. [14] for the needed background in algebraic topology. Whenever the added cell has dimension p, it either gives *birth* to a degree-p homology class or it gives *death* to a degree-$(p-1)$ homology class. This is classic Morse theory, and the novelty in persistence is that for each death-giving $(p+1)$-cell we find a unique birth-giving p-cell whose class it ends. We call these the degree-p *birth-death pairs* of f, denoted $BD_p(f)$. It is also possible that some birth-giving p-cells do not get paired, and they reflect the degree-p homology of K. A convenient combinatorial representation of this information is the *persistence diagram*, denoted $\mathrm{Dgm}_p(f)$, which maps $(a,b) \in BD_p(f)$ to the point $(f(a), f(b)) \in \mathbb{R}^2$.

An important property is the stability of the persistence diagram, originally proved in [6]. In a nutshell, it says that for two monotonic functions, $f, g\colon K \to \mathbb{R}$, there is a bijection between $\mathrm{Dgm}_p(f)$ and $\mathrm{Dgm}_p(g)$ such that the maximum coordinate difference between corresponding points is bounded from above by the absolute maximum of $f(a) - g(a)$ over all cells $a \in K$. This will be instrumental in the following construction.

Consider two filters, $f_0, f_1 \colon K \to \mathbb{R}$, and the straight-line homotopy defined by $f_\lambda(a) = (1-\lambda)f_0(a) + \lambda f_1(a)$ for $0 \leq \lambda \leq 1$. These functions imply a 1-parameter family of persistence diagrams for each dimension p. Mapping every point $(f_\lambda(a), f_\lambda(b)) \in \mathrm{Dgm}_p(f_\lambda)$ to $(f_\lambda(a), f_\lambda(b), \lambda) \in \mathbb{R}^3$, we effectively stack up these diagrams, and because of stability, the points form continuous curves in $\mathbb{R}^3$. Following the terminology in [7], we call these curves *vines* and the collection of vines a *vineyard*. Assuming K is finite, there are only finitely many values of λ for which f_λ is not a filter (because it assigns the same value to at least two cells). These values split $[0,1]$ into finitely many open intervals with a unique ordering of the cells in each. In the generic case, the orderings in any two consecutive intervals differ by a single *transposition* of two adjacent cells. Such a transposition may or may not affect the birth-death pairs, and if it does, we call it a *switch*. As established in [7], there are only three kinds of switches: the *BB-type*, which swaps the birth-giving cells between two pairs, the *DD-type*, which swaps the death-giving cells between two pairs, and the *BD-type*, which swaps a birth-giving with a death-giving cell; see Figure 2. Observe that in the first two cases, the two affected points belong to the same-degree persistence diagram, while in the third case the affected points belong to persistence diagrams of consecutive degrees.

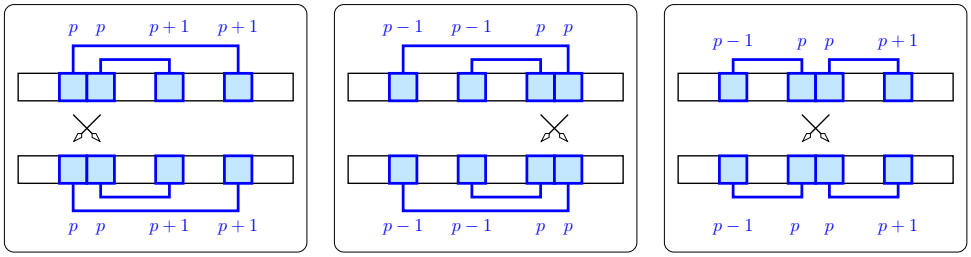


Figure 2 The three types of switches. *From left to right:* swapping two birth-giving cells (BB-type), swapping two death-giving cells (DD-type), and swapping a birth-giving with a death-giving cell (BD-type). In the first two cases, the intervals that correspond to the pairs are nested – before and after the switch – and in the third case, they are disjoint – again before and after the switch.

Cancellation of Shallow Pairs. Given a filtered Lefschetz complex, we are interested in simplifying it. The natural operation to this end is the *cancellation* of a cell b with one of its facets a. The operation produces a new Lefschetz complex $(K', \dim', \kappa')$ where $K' = K \setminus \{a, b\}$, $\dim'$ is the restriction of dim to K', and the new incidence map is defined for $x, y \in K'$ by $\kappa'(y, x) = \kappa(y, x) + \kappa(y, a) \cdot \kappa(b, x)$; see the schematic cancellation of the pair (a, b) in Figure 3.

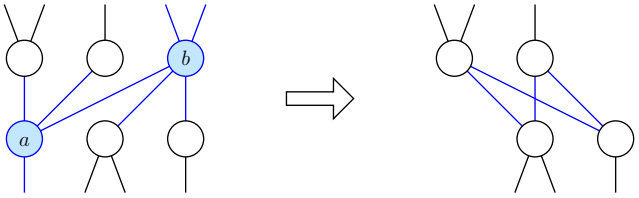


Figure 3 To cancel $a \prec b$, we connect all facets of b to all cofacets of a. The remainder of the face relation is unchanged.

A cancellation may have side-effects, such as changing the homotopy type of the complex or making a cell a facet of another cell that precedes it in the filter. In contrast, a cancellation preserves the homology of the Lefschetz complex [9]. To avoid the need to reorganize the

filter, we restrict ourselves to cancellations where this is not necessary. Even more, we restrict ourselves to canceling *shallow pairs*, $a \prec b$, for which a is the last facet of b and b is the first cofacet of a in the ordering of the filter; see [9] but also [2], where they are referred to as *apparent pairs*. If $a \prec b$ is a shallow pair, then all other facets of b precede a and all other cofacets of a succeed b, so the new filter is obtained by removing a and b, and no reordering is necessary. The following basic properties are obvious or proved in [9]:

- every shallow pair is a birth-death pair of the filter;
- after a shallow pair is canceled, the remaining birth-death pairs are unchanged;
- there is at least one shallow pair, unless the face relation is empty.

Because of the third property, we can keep canceling shallow pairs until the face relation runs empty. We call the resulting sequence of canceled pairs a *shallow cancellation order*, which by the first two properties is a total order of the birth-death pairs of the original filter. However, there may be many orderings of the birth-death pairs such that each pair is shallow at the time it gets canceled, which motivates the next concept.

▶ **Definition 2.2.** *The* depth poset *of a filter on a Lefschetz complex, $f\colon K \to \mathbb{R}$, is the intersection of all shallow cancellation orders of f, denoted* $\mathrm{Depth}(f)$.

This concept was introduced in [9], along with some of its fundamental properties, which we state without proof.

▶ **Lemma 2.3.** *Let $f\colon K \to \mathbb{R}$ be a filter on a Lefschetz complex, $\varphi = (x, y)$ and $\psi = (a, b)$ two birth-death pairs of f, (φ, ψ) an arc in* $\mathrm{Depth}(f)$, *$\sigma = (s, t)$ a shallow pair of f, and $f'\colon K \setminus \{s, t\}$ the filter after canceling σ. Then*

1. *nodes connected by an arc in* $\mathrm{Depth}(f)$ *are nested; that is: $f(a) < f(x) < f(y) < f(b)$;*
2. *arcs of* $\mathrm{Depth}(f)$ *respect dimensions; that is:* $\dim a = \dim x$ *and* $\dim b = \dim y$;
3. *canceling σ has no side-effects; that is:* $\mathrm{Depth}(f') = \mathrm{Depth}(f) \cap [BD(f') \times BD(f')]$.

By Property 2, $\mathrm{Depth}(f)$ is the disjoint union of posets $\mathrm{Depth}_p(f) \subseteq BD_p(f) \times BD_p(f)$.

Matrix Reduction Twice. While the definition of the depth poset is constructive, it is in terms of possibly exponentially many total orders on the birth-death pairs. Alternatively, it can be constructed as the transitive closure of two relations, both of which can be computed efficiently by reducing the boundary matrix of the complex. We review the algorithms and refer to [9] for a proof of correctness.

▶ **Definition 2.4.** *The* death relation *of a filter on a Lefschetz complex, $f\colon K \to \mathbb{R}$, denoted $D(f) \subseteq BD(f) \times BD(f)$, records the left-to-right column operations during the reduction of the boundary matrix in reverse order of the birth-giving cells; see Algorithm 1.*

The arcs in the death relation correspond to the off-diagonal non-zero entries in the columns of death-giving cells in U_1:

$$D(f) = \{((x, y), (a, b)) \in BD(f) \times BD(f) \mid y \neq b \text{ and } U_1[y, b] = 1\}. \tag{2}$$

To make sense of U_1, consider the matrix V_1 that stores the chains whose boundaries are the cycles in R_1. Initially, V_1 is the identity matrix and we maintain it by applying the same column operations as performed on R_1. If we ignore the deletions of rows and columns, we retain the matrices in their original sizes, and it is not difficult to see that they satisfy $R_1 = \Delta V_1$ and $\Delta = R_1 U_1$ throughout the algorithm, so $U_1 = V_1^{-1}$.

▶ **Definition 2.5.** *The* birth relation *of a filter on a Lefschetz complex, $f\colon K \to \mathbb{R}$, denoted $B(f) \subseteq BD(f) \times BD(f)$, records the bottom-to-top row operations during the reduction of the boundary matrix in the order of the death-giving cells; see Algorithm 2.*

Algorithm 1 Bottom to Top Column Reduction.

Input: Δ (the boundary matrix)
Output: R_1U_1 (a decomposition of Δ)

$R_1 = \Delta$; $U_1 = \mathrm{Id}$;
while $R_1 \neq 0$ **do**
  let $R_1[x, y]$ be the leftmost non-zero entry in the lowest non-zero row;
  while $\exists b > y$ such that $R_1[x, b] = 1$ **do**
    add column y to column b in R_1; $U_1[y, b] = U_1[y, b] + 1$
  end while;
  delete row x and column y from R_1
end while.

Algorithm 2 Left to Right Row Reduction.

Input: Δ (the boundary matrix)
Output: R_2U_2 (a decomposition of Δ)

$R_2 = \Delta$; $U_2 = \mathrm{Id}$;
while $R_2 \neq 0$ **do**
  let $R_2[x, y]$ be the lowest non-zero entry in the leftmost non-zero column;
  while $\exists a < x$ such that $R_2[a, y] = 1$ **do**
    add row x to row a in R_2; $U_2[a, x] = U_2[a, x] + 1$
  end while;
  delete row x and column y from R_2
end while.

The arcs in the birth relation correspond to the off-diagonal non-zero entries in the rows of birth-giving cells in U_2:

$$B(f) = \{((x, y), (a, b)) \in BD(f) \times BD(f) \mid x \neq a \text{ and } U_2[a, x] = 1\}. \tag{3}$$

To rationalize U_2, we introduce the matrix V_2 that stores the chains whose coboundaries are the cocycles, i.e. $R_2 = V_2\Delta$ assuming we ignore the deletions of rows and columns. Similarly $\Delta = U_2R_2$, so $U_2 = V_2^{-1}$.

The right panel of Figure 1 shows examples of the death and birth relations of a filter. A noteworthy feature of this example are the pairs $(\mathtt{d}, \gamma)$ and (β, Σ). In spite of the fact that $\mathtt{d}$ is an edge of β, which prevents $(\mathtt{d}, \gamma)$ from being shallow, there is no arc connecting the two pairs in the depth poset. Indeed, the two pairs are neither nested nor do their dimensions match, so an arc would contradict the first two properties in Lemma 2.3. Hence, it must be possible to cancel $(\mathtt{d}, \gamma)$ before canceling (β, Σ), and this can indeed be done if we first cancel $(\mathtt{e}, \alpha)$, which is the sole predecessor of $(\mathtt{d}, \gamma)$ in the poset. Geometrically, the cancellation of $(\mathtt{e}, \alpha)$ attaches β to $\mathtt{d}$ twice, which in modulo-2 arithmetic is counted as no attachment.

It is not difficult to see that the two relations are neither necessarily their own transitive closures nor necessarily their own transitive reductions, and while they express different pieces of information, they are not necessarily disjoint. Nevertheless, they determine the depth poset as the transitive closure of their union; see [9] for a proof. This is important so we state the claim more formally.

▶ **Theorem 2.6.** *Let $f\colon K \to \mathbb{R}$ be a filter on a Lefschetz complex. Then* $\mathrm{Depth}(f)$ *is the transitive closure of $D(f) \cup B(f)$.*

3 Case Analysis

A transposition preserves all birth-death pairs, unless it is a switch, in which case the two transposed cells replace each other in their respective pairs. Consider for example the filter in Figure 1 and entertain what happens when we transpose the edges c and d. Comparing Figures 1 and 4, we note that (C, c) and (d, γ) got replaced by (C, d) and (c, γ), and the death and birth relations both lost their only arc. It should however be noted that a transposition

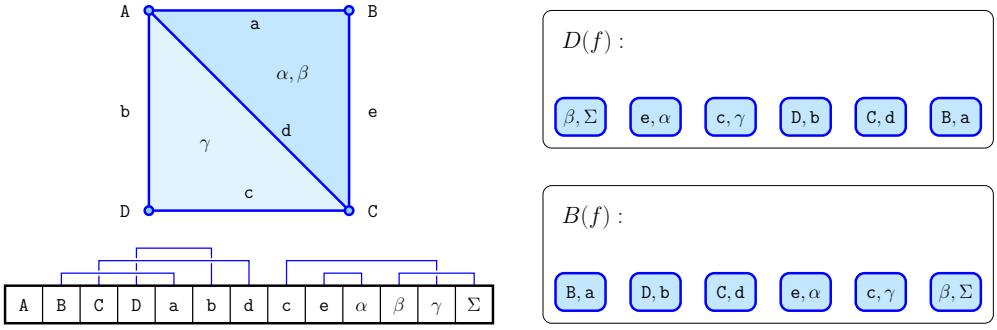


Figure 4 The only difference to the example in Figure 1 is the order of the edges c and d. To the *right*, we see the death and birth relations, which are both empty.

may change the depth poset without affecting the birth-death pairs. To see this, we organize a complete case analysis along the classification of transpositions into three types: of two birth-giving cells, of two death-giving cells, and of a birth-giving with a death-giving cell.

Prior to the case analysis, we state two lemmas that will both be repeatedly used. The first excludes certain configurations of birth-death pairs.

▶ **Lemma 3.1.** *Let $f\colon K \to \mathbb{R}$ be a filter on a Lefschetz complex, and $a \prec y$ two cells in K. If a is the last facet of y in the ordering implied by f, and a is not yet paired when we encounter y in the ordering of the filter, then (a, y) is a birth-death pair of f.*

Proof. Consider the standard matrix reduction algorithm for persistent homology, which reduces the columns of the ordered boundary matrix from left to right. Since a is the last facet of y, the lowest non-zero entry in column y is in row a. When the algorithm arrives at y, a is not yet paired, so there is no pivot in its row. It follows that this entry remains and becomes a pivot, which is equivalent to (a, y) becoming a birth-death pair. ◀

The second lemma addresses a property of ordered boundary matrices that amounts to a reformulation of Lemma 4.6 in [9]. We state it without proof.

▶ **Lemma 3.2.** *Let $f\colon K \to \mathbb{R}$ be a filter on a Lefschetz complex, Δ the corresponding ordered boundary matrix, x a row of Δ, and y a column of Δ. Then any two boundary matrices obtained by canceling all birth-death pairs with pivots below row x or to the left of column y following shallow but possibly different cancellation orders are the same.*

3.1 Case I: Birth-birth Transpositions

Let (x, y) and (a, b) be two nested or otherwise overlapping birth-death pairs with matching dimension, so $f(a) < f(x) < f(y) < f(b)$ or $f(x) < f(a) < f(y) < f(b)$ and $\dim a = \dim x = \dim y - 1 = \dim b - 1$. As illustrated in Figure 5, we assume that a and x are consecutive in the ordering, so either the two pairs are incomparable in the depth poset, or they are

connected by an arc but not by a path of two arcs. To prepare the analysis, we cancel the birth-death pairs whose pivots are below row x or to the left of column y in the ordered boundary matrix. By Lemma 3.2, the sequence in which these pairs are canceled does not affect the remaining portion of the boundary matrix. After the cancellations, (x, y) is shallow and (a, b) is either shallow or has (x, y) as its sole predecessor in the depth poset. Similarly, we cancel all birth-death pairs with pivots below row a or to the left of column y in the boundary matrix after transposing a and x. The two operations commute, so we can alternatively transpose a and x after the mentioned cancellations. Refer to Figure 5 for the three sub-cases as defined after the initial cancellations:

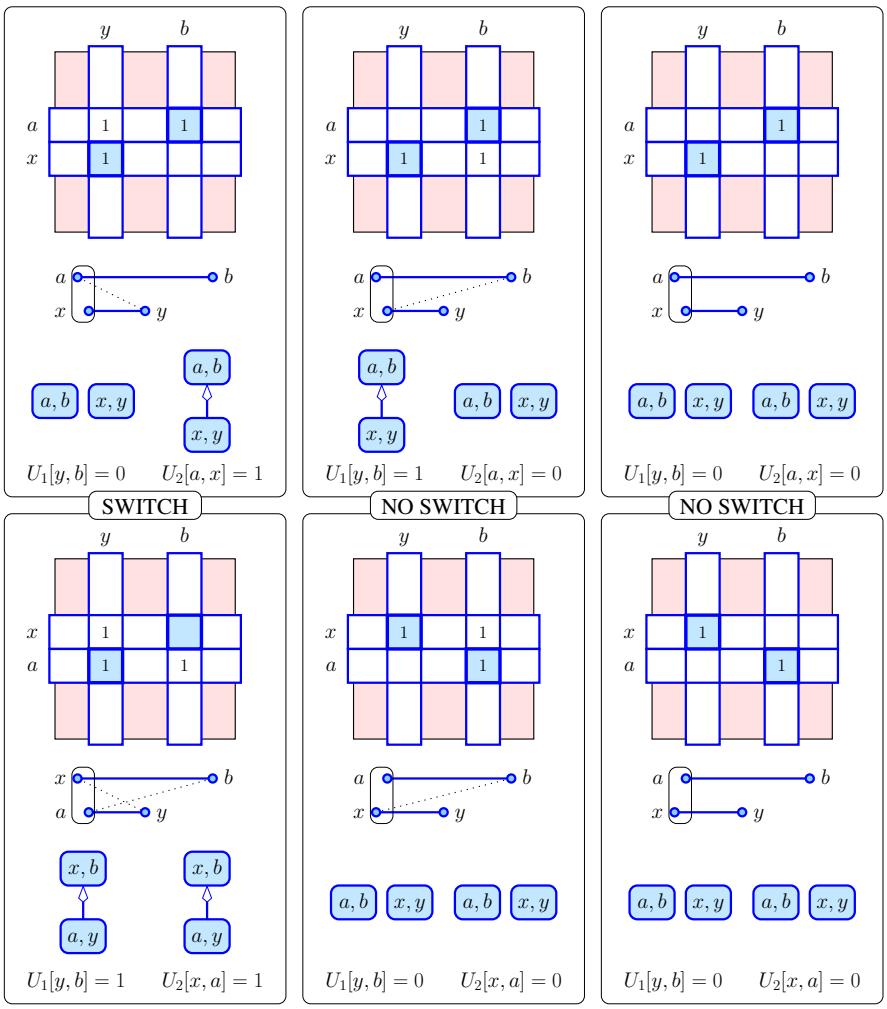


Figure 5 The configurations before and after the transposition of two birth-giving cells. *From left to right:* Cases I.1, I.2, I.3, with the configuration before and after the transposition at the *top* and the *bottom*, respectively, or vice versa for the inverse transposition. Within each panel, we show the relevant birth- and death-giving cells as rows and columns of the boundary matrix, respectively, the two birth-death pairs as bars with *dotted* face relation, and the corresponding nodes and arcs in the death and birth relations.

Case I.1: $a \prec y$ and $x \not\prec b$, as illustrated in the upper left panel of Figure 5. Since (a, b) is not shallow, it must be that (x, y) precedes (a, b) in the depth poset before the transposition of a and x. To see that the transposition is a switch note that after transposing, a is

the last facet of y and is not yet paired before we encounter y from the left, so (a, y) is a birth-death pair by Lemma 3.1. By the stability of the persistence diagram [6], (x, b) is another pair after the transposition, and the remaining pairs are unchanged; see the lower left panel of Figure 5. Finally, reading the two panels from bottom to top, we have the inverse switch defined by $x \prec y$ and $a \prec b$.

Case I.2: $a \not\prec y$ and $x \prec b$, as illustrated in the upper middle panel of Figure 5. Like in Case I.1, (x, y) is a predecessor of (a, b) before the transposition of a and x. Since a is not a facet of y, (x, y) remains shallow, so the transposition is not a switch. But since the two pairs are no longer nested, they are also no longer comparable by Lemma 2.3; see the lower middle panel of Figure 5. Reading the two panels from bottom to top, we have the inverse transposition defined by $a \not\prec y$ and $x \prec b$, but now for two overlapping pairs.

Case I.3: $a \not\prec y$ and $x \not\prec b$, as illustrated in the upper right panel of Figure 5. Like in Case I.2, the transposition of a and x maintains (x, y) as a shallow pair, so the transposition is not a switch. The two pairs are incomparable before the transposition because x is not a facet of b and y is not a cofacet of a, and after the transposition because the pairs are no longer nested; see the lower right panel of Figure 5. The inverse transposition is defined by $a \not\prec y$ and $x \not\prec b$, but now for pairs that are neither nested nor disjoint.

When the transposition of a and x is not a switch, then the changes described in Cases I.2 and I.3 are exhaustive, but if it is a switch, there may be changes beyond those described in Case I.1. We need notation to describe them. Letting $\varphi = (x, y)$ and $\psi = (a, b)$ be two birth-death pairs, the former is a predecessor of the latter in the death relation if $U_1[y, b] = 1$ and in the birth relation if $U_2[a, x] = 1$. We therefore write

$$\mathrm{Pred}_1(a,b) = \{\varphi \mid (\varphi,\psi) \in D\} = \{(x,y) \in BD(f) \mid y \neq b \text{ and } U_1[y,b] = 1\}; \tag{4}$$
$$\mathrm{Pred}_2(a,b) = \{\varphi \mid (\varphi,\psi) \in B\} = \{(x,y) \in BD(f) \mid a \neq x \text{ and } U_2[a,x] = 1\}; \tag{5}$$
$$\mathrm{Succ}_1(a,b) = \{\varphi \mid (\psi,\varphi) \in D\} = \{(x,y) \in BD(f) \mid b \neq y \text{ and } U_1[b,y] = 1\}; \tag{6}$$
$$\mathrm{Succ}_2(a,b) = \{\varphi \mid (\psi,\varphi) \in B\} = \{(x,y) \in BD(f) \mid x \neq a \text{ and } U_2[x,a] = 1\}, \tag{7}$$

and $\mathrm{Pred}_1^{\mathrm{bef}}(a,b)$, $\mathrm{Pred}_1^{\mathrm{aft}}(a,b)$ for the sets before and after a transposition, etc. We begin with the changes of the successors in the death relation and write $\oplus$ for the symmetric difference operation between sets.

▶ **Lemma 3.3.** *Let $f\colon K \to \mathbb{R}$ be a filter on a Lefschetz complex, (x, y) and (a, b) birth-death pairs of matching dimension, a precedes x, y precedes b, and a, x are consecutive in the ordering implied by the filter. Then $a \prec y$ iff $(x, y) \in \mathrm{Pred}_2(a, b)$ iff the transposition of x and a is a BB-type switch (see the left panels in Figure 5), and in this case*

$$\mathrm{Succ}_1^{\mathrm{aft}}(a,y) = \mathrm{Succ}_1^{\mathrm{bef}}(x,y) \oplus (x,b) \oplus \mathrm{Succ}_1^{\mathrm{bef}}(a,b); \quad \mathrm{Succ}_1^{\mathrm{aft}}(x,b) = \mathrm{Succ}_1^{\mathrm{bef}}(a,b); \tag{8}$$
$$\mathrm{Succ}_2^{\mathrm{aft}}(a,y) = \mathrm{Succ}_2^{\mathrm{bef}}(x,y) \oplus \{(x,b),(a,b)\}; \quad \mathrm{Succ}_2^{\mathrm{aft}}(x,b) = \mathrm{Succ}_2^{\mathrm{bef}}(a,b); \tag{9}$$
$$\mathrm{Pred}_1^{\mathrm{aft}}(a,y) = \mathrm{Pred}_1^{\mathrm{bef}}(x,y); \quad \mathrm{Pred}_1^{\mathrm{aft}}(x,b) = \mathrm{Pred}_1^{\mathrm{bef}}(a,b); \tag{10}$$
$$\mathrm{Pred}_2^{\mathrm{aft}}(a,y) = \mathrm{Pred}_2^{\mathrm{bef}}(x,y); \quad \mathrm{Pred}_2^{\mathrm{aft}}(x,b) = \mathrm{Pred}_2^{\mathrm{bef}}(a,b). \tag{11}$$

Otherwise, the depth relation remains as is, except if $x \prec b$ (see the middle panels in Figure 5), in which case $\mathrm{Succ}_1^{\mathrm{aft}}(x,y) = \mathrm{Succ}_1^{\mathrm{bef}}(x,y) \oplus (a,b) \oplus \mathrm{Succ}_1^{\mathrm{bef}}(a,b)$, while all other sets of successors and predecessors of (x, y) and (a, b) remain unchanged.

Proof. The two equivalences follow from the completeness of the case analysis displayed in Figure 5. To see (8), consider rows a and x in the boundary matrix before and after the switch. With reference to the upper left panel of Figure 5 – which shows the matrix before the switch – we give names to the portions of these rows delimited by columns y and b:

$$M_a = \{R_1[a,t] \mid f(y) \le f(t) < f(b)\}; \quad N_a = \{R_1[a,t] \mid f(b) \le f(t)\}; \tag{12}$$
$$M_x = \{R_1[x,t] \mid f(y) \le f(t) < f(b)\}; \quad N_x = \{R_1[x,t] \mid f(b) \le f(t)\}. \tag{13}$$

Canceling (x,y) changes row a to $(M_a + M_x)|(N_a + N_x)$, in which the bar means catenation. At this time, (a,b) is a shallow pair, so $M_a + M_x = 0$, which implies $M_a = M_x$. Columns y and b in U_1 are transposed copies of rows x and a at the moment before (x,y) and then (a,b) are canceled. Hence, column y in U_1 is $(M_x|N_x)^T$, and column b is $(N_a + N_x)^T$. This is before the transposition of a and x. After the transposition, column y in U_1 is $(M_a|N_a)^T$, and column b is $(N_x + N_a)^T$. Using $M_a = M_x$ and $N_a = N_x + (N_a + N_x)$, we rewrite these relations in different notation:

$$U_1^{\text{aft}}[y,t] = \begin{cases} U_1^{\text{bef}}[y,t] & \text{for } f(y) \le f(t) < f(b), \\ U_1^{\text{bef}}[y,t] + U_1^{\text{bef}}[b,t] & \text{for } f(b) \le f(t); \end{cases} \tag{14}$$
$$U_1^{\text{aft}}[b,t] = U_1^{\text{bef}}[b,t]. \tag{15}$$

Finally, we translate the relations into the notation of the lemma using (6), and get (8) from (14) and (15). We omit the arguments for the remaining relations in Lemma 3.3. ◀

The symmetric cases – when x precedes a so the birth-birth transpositions go from the lower to the upper panes in Figure 5 – are similar and the same relations between the successors and predecessors before and after the transposition apply.

3.2 Case II: Death-death Transpositions

The setting is similar to Case I: (x,y) and (a,b) are nested or otherwise overlapping birth-death pairs of matching dimension, but now we assume that y and b are consecutive in the ordering implied by the filter. As before, we prepare the analysis by canceling all birth-death pairs whose pivots are below row x or to the left of column y. After these cancellations, (x,y) is shallow, and (a,b) is shallow or it has (x,y) as its sole predecessor in the depth poset. Refer to Figure 6 for the three sub-cases as defined after the initial cancellations:

Case II.1: $a \not\prec y$ and $x \prec b$ for a nested pair, as illustrated in the upper left panel of Figure 6. The transposition is a switch, and reading the two left panels from bottom to top, we have the inverse switch defined by $x \prec y$ and $a \prec b$.

Case II.2: $a \prec y$ and $x \not\prec b$ for a nested pair, as illustrated in the upper middle panel of Figure 6. The transposition is not a switch, and reading the two middle panels from bottom to top, we have the inverse transposition defined by the same two conditions for overlapping pairs.

Case II.3: $a \not\prec y$ and $x \not\prec b$ for a nested pair, as illustrated in the upper right panel of Figure 6. Again this transposition is not a switch, and reading the two right panels from bottom to top, we have the inverse transposition defined by the same two conditions for overlapping pairs.

The sub-cases are strictly symmetric to Cases I.1, I.2, I.3, with identical arguments for the symmetric actions. While omitting other details, we state the result symmetric to Lemma 3.3, which describes the changes to the death and birth relations beyond the pairs that contain the transposed cells.

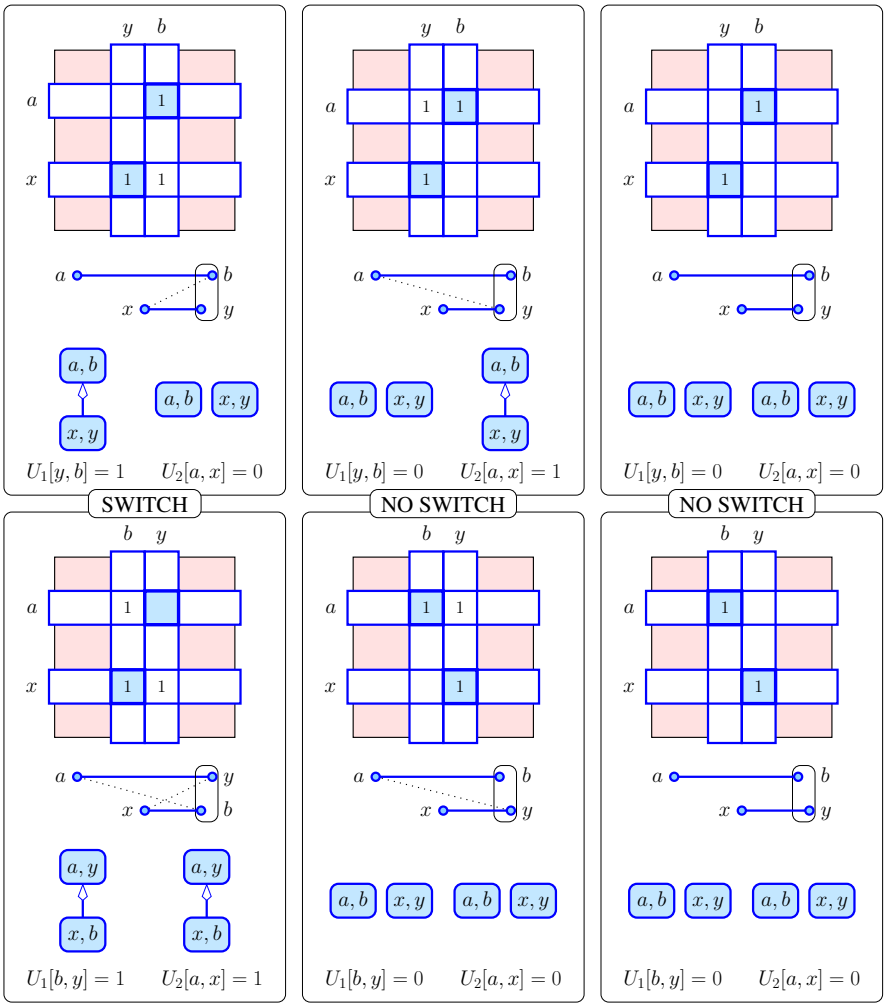


Figure 6 The configurations before and after the transposition of two death-giving cells in the filter. *From left to right:* Cases II.1, II.2, II.3, with the configuration before the transposition at the *top* and after the transposition at the *bottom*, or vice versa for the inverse transposition. Within each panel, we show the relevant birth-giving cells as rows and death-giving cells as columns of the matrix, the two birth-death pairs as bars, and the corresponding nodes and relations in the death and birth relations.

▶ **Lemma 3.4.** *Let $f\colon K \to \mathbb{R}$ be a filter on a Lefschetz complex, (x,y), (a,b) birth-death pairs of matching dimension, a precedes x, y precedes b, and y,b are consecutive in the ordering implied by the filter. Then $x \prec b$ iff $(x,y) \in \mathrm{Pred}_1(a,b)$ iff the transposition of y and b is a DD-type switch (see the left panels in Figure 6). and in this case*

$$\mathrm{Succ}_1^{\mathrm{aft}}(x,b) = \mathrm{Succ}_1^{\mathrm{bef}}(x,y) \oplus \{((a,y),(a,b)\}; \qquad \mathrm{Succ}_1^{\mathrm{aft}}(a,y) = \mathrm{Succ}_1^{\mathrm{bef}}(a,b); \quad (16)$$

$$\mathrm{Succ}_2^{\mathrm{aft}}(x,b) = \mathrm{Succ}_2^{\mathrm{bef}}(x,y) \oplus (a,y) \oplus \mathrm{Succ}_2^{\mathrm{bef}}(a,b); \quad \mathrm{Succ}_2^{\mathrm{aft}}(a,y) = \mathrm{Succ}_2^{\mathrm{bef}}(a,b); \quad (17)$$

$$\mathrm{Pred}_1^{\mathrm{aft}}(x,b) = \mathrm{Pred}_1^{\mathrm{bef}}(x,y); \qquad \mathrm{Pred}_1^{\mathrm{aft}}(a,y) = \mathrm{Pred}_1^{\mathrm{bef}}(a,b); \quad (18)$$

$$\mathrm{Pred}_2^{\mathrm{aft}}(x,b) = \mathrm{Pred}_2^{\mathrm{bef}}(x,y); \qquad \mathrm{Pred}_2^{\mathrm{aft}}(a,y) = \mathrm{Pred}_2^{\mathrm{bef}}(a,b). \quad (19)$$

Otherwise, the depth relation remains as is, except if $a \prec y$ (see the middle panels in Figure 6), in which case $\mathrm{Succ}_2^{\mathrm{aft}}(x,y) = \mathrm{Succ}_2^{\mathrm{bef}}(x,y) \oplus (a,b) \oplus \mathrm{Succ}_2^{\mathrm{bef}}(a,b)$, while all other sets of sets of successors and predecessors of (s,y) and (a,b) remain unchanged.

3.3 Case III: Birth-death Transpositions

Case III is characterized by the transposition of cells b and x, in which (a, b) and (x, y) are disjoint or overlapping birth-death pairs with consecutive dimensions, so $f(a) < f(b) < f(x) < f(y)$ or $f(a) < f(x) < f(b) < f(y)$ and $\dim a + 1 = \dim b = \dim x = \dim y - 1$. Similar to Cases I and II, we prepare the analysis by canceling all pairs whose pivots lie below the rows or to the left of columns b and x. This includes all predecessors of (a, b) and (x, y), so after these cancellations, both are shallow. Refer to Figure 7 for the two sub-cases, with the conditions referring to the complex after the initial cancellations:

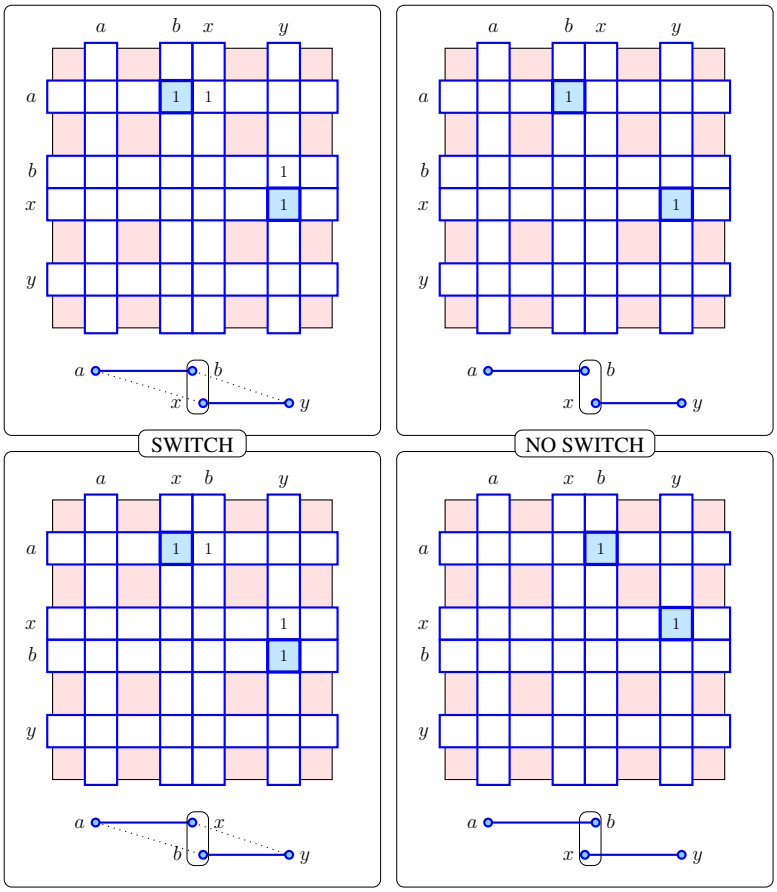


Figure 7 The configurations in Cases III.1 and III.2 on the *left* and *right*, before and after the transposition of a birth-giving cell with a death-giving cell. Within each panel, we show the relevant rows and columns of the matrix, and the corresponding birth-death pairs as bars.

Case III.1: $a \prec x$ and $b \prec y$ for nested pairs, as illustrated in the upper left panel of Figure 7. After transposing b and x, (a, x) and (b, y) are both shallow, so the transposition is a switch and the two new pairs are birth-death pairs of the new filter. By the first two properties in Lemma 2.3, there is no arc connecting the two pairs in the depth poset. The inverse transposition is symmetric and follows the same rules.

Case III.2: $a \not\prec x$ and $b \not\prec y$ for nested pairs, as illustrated in the upper right panel of Figure 7. After transposing b and x, (a, b) and (x, y) are still shallow, so the transposition is not a switch. The inverse transposition is defined by the same conditions but now for two overlapping pairs, as illustrated in the bottom right panel of Figure 7.

There are no additional cases, because the relevant birth-death pairs are shallow both before and after the transposition, which excludes configurations different from the ones illustrated in Figure 7. Observe that Case III.1 is quite different from all others, because the transposition changes the types of two cells: b from death- to birth-giving, and x from birth- to death-giving. Hence, their corresponding rows and columns in the boundary matrix assume different roles, which may have a profound impact on the depth poset. We prove that this is in fact not the case for the successors, but it can be for the predecessors.

Recall that Algorithm 1 records the column reductions for all cells alike, but only the ones for death-giving cells turn into arcs of the death relation. Symmetrically, Algorithm 2 records the row reductions for all cells, but only the ones for birth-giving cells that also belong to birth-death pairs turn into arcs of the birth relation. We will now make use of the so far neglected information. Letting a and x be the birth-giving cells of the two pairs in a BD-type switch, we collect the reductions of column x prior to reaching row a as arcs in the death relation, and letting y and b be death-giving cells of these pairs, we collect the reductions of row b prior to reaching column y as arcs in the birth relation.

▶ **Lemma 3.5.** *Let $f\colon K \to \mathbb{R}$ be a filter on a Lefschetz complex, (a,b) and (x,y) birth-death pairs of consecutive dimensions, and b,x consecutive in the ordering implied by the filter. Then $a \prec x$ and $b \prec y$ iff the transposition of b and x is a BD-type switch (see the left panels in Figure 7), and in this case*

$$\mathrm{Succ}_1^{\mathrm{aft}}(a,x) = \mathrm{Succ}_1^{\mathrm{bef}}(a,b); \qquad \mathrm{Succ}_1^{\mathrm{aft}}(b,y) = \mathrm{Succ}_1^{\mathrm{bef}}(x,y); \tag{20}$$

$$\mathrm{Succ}_2^{\mathrm{aft}}(a,x) = \mathrm{Succ}_2^{\mathrm{bef}}(a,b); \qquad \mathrm{Succ}_2^{\mathrm{aft}}(b,y) = \mathrm{Succ}_2^{\mathrm{bef}}(x,y); \tag{21}$$

$$\mathrm{Pred}_1^{\mathrm{aft}}(a,x) = \{(t,x) \mid U_1^{\mathrm{bef}}[t,x] = 1, f(t) > f(a)\}; \quad \mathrm{Pred}_1^{\mathrm{aft}}(b,y) = \mathrm{Pred}_1^{\mathrm{bef}}(x,y); \tag{22}$$

$$\mathrm{Pred}_2^{\mathrm{aft}}(a,x) = \mathrm{Pred}_2^{\mathrm{bef}}(a,b); \quad \mathrm{Pred}_2^{\mathrm{aft}}(b,y) = \{(b,s) \mid U_2^{\mathrm{bef}}[b,s] = 1, f(s) < f(y)\}. \tag{23}$$

Otherwise, the depth relation remains as is (see the right panels in Figure 7).

Proof. To prove (20) and (21), we first consider rows b and x in the boundary matrix (after canceling all birth-death pairs whose pivots are below row x or to the left of column b). Before the transposition, b gives death, so there is no pivot in its row. We claim that this implies that the two rows are equal. Indeed, if there is a column in which the entries in rows b and x are different, then the leftmost such entry would be a pivot, which is a contradiction to the assumptions. Since the two rows are the same, transposing them makes no difference, other than exchanging their names. This implies $\mathrm{Succ}_1^{\mathrm{aft}}(a,x) = \mathrm{Succ}_1^{\mathrm{bef}}(a,b)$ as well as $\mathrm{Succ}_1^{\mathrm{aft}}(b,y) = \mathrm{Succ}_1^{\mathrm{bef}}(x,y)$. The symmetric argument implies the corresponding relations for the successors in the birth relation.

To prove (22) and (23), we recall that rows b and x are equal, so in the death relation the predecessors of (x,y) before the transposition are the predecessors of (b,y) after the transposition and, symmetrically, in the birth relation the predecessors of (a,b) before the transposition are the predecessors of (a,x) after the transposition. The remaining predecessors of (a,x) and (b,y) are determined while canceling the birth-death pairs with pivots below and to the left of the rows and columns of these pairs. We can get this information from column x in U_1 and row b in U_2 as computed before the transposition. ◀

The symmetric cases – in which we read the panels in Figure 7 from bottom to top – are similar. This completes the case analysis of changes to the depth poset caused by the transposition of two cells that are consecutive in the filter.

4 Computational Experiments

In this section, we present measurements of the depth poset collected while running the algorithms on 1-, 2-, and 3-dimensional random functions and straight-line homotopies between them. For each set of parameters, the results are averaged over ten repeats.

4.1 Random Piecewise Linear Functions

To minimize complications and avoid boundary effects as well as any essential homology, we construct the random functions on a regular cubical subdivision of the d-dimensional torus. Choosing a positive integer, n, we use $(\mathbb{R}/n\mathbb{Z})^d$ as a model of the d-torus, and subdivide it with the unit d-cubes of $(\mathbb{Z}/n\mathbb{Z})^d + [0,1]^d$. We furthermore kill its essential homology classes by adding 2^d *extra cells*, namely the empty set as the sole (-1)-cell and $\binom{d}{p}$ $(p+1)$-cells, each with boundary equal to a subdivided p-torus, for $1 \leq p \leq d$. Write $K = K(n,d)$ for the resulting complex, which consists of $\binom{d}{p}n^d$ p-cubes, for $0 \leq p \leq d$, plus the 2^d extra cells.

As an example, let $d = 2$ in which case the torus is the n-by-n square grid, which is subdivided into n^2 unit square cells. The (-1)-cell facilitates reduced homology, which pairs the empty set with the component. In addition, there are two 2-cells killing the essential degree-1 homology classes, each a polygon whose boundary consists of n edges, a horizontal integer line for one and a vertical integer line in the square grid for the other polygon. Finally, there is the 3-cell whose boundary is the entire n-by-n square grid.

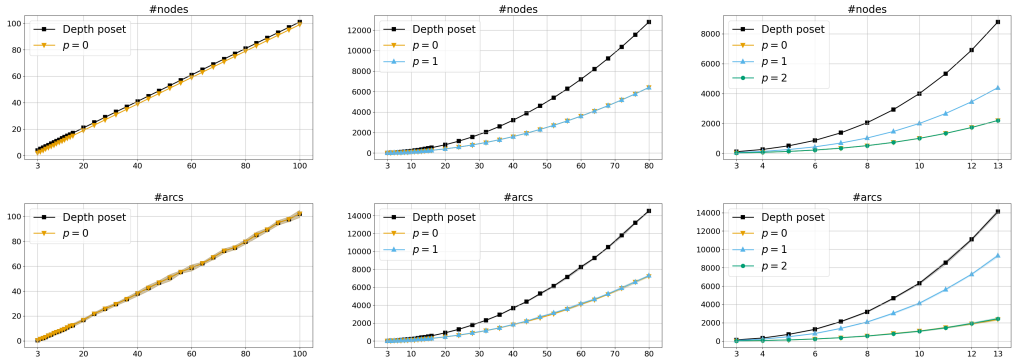


Figure 8 *From left to right:* the average number of nodes (*upper row*) and arcs (*lower row*) in the transitive reduction of the depth poset of a random function on the d-torus, for $d = 1, 2, 3$, respectively. Because of the symmetries in the construction, the curves for $p = 0$ and $p = d - 1$ are nearly identical.

To turn this complex into a random function, $f\colon K \to \mathbb{R}$, we choose $f(\gamma)$ in $[p, p+1]$, whenever $p = \dim \gamma$. In particular, we choose $f(\gamma)$ uniformly at random in $[p, p+1]$ whenever γ is a p-cube, and we set $f(\gamma) = p+1$ whenever it is an extra p-cell. In spite of the extra cells that kill the essential homology of the d-torus, we will refer to f as a *random function on the d-torus*. By construction, the filter that orders the cells by value is also ordered by dimension. We may think of f as a piecewise constant function, but there is an alternative interpretation as a piecewise linear function. Letting $\hat{K} = \hat{K}(n,d)$ be the barycentric subdivision of K, each vertex $\hat{\gamma} \in \hat{K}$ is the center of a cell $\gamma \in K$, and we map $\hat{\gamma}$ to $\hat{f}(\hat{\gamma}) = f(\gamma)$. We then get $\hat{f}$ on the underlying space of K by piecewise linear interpolation of its values at the vertices. By construction, each center of a p-cell in K is a critical point of index p of $\hat{f}$. All critical points are paired up as nodes of the depth poset, which implies that n and d determine the number of nodes for each $-1 \leq p \leq d$; see the upper row in Figure 8. In contrast, the number of arcs in the depth poset has a small but non-zero variance; see the lower row of Figure 8.

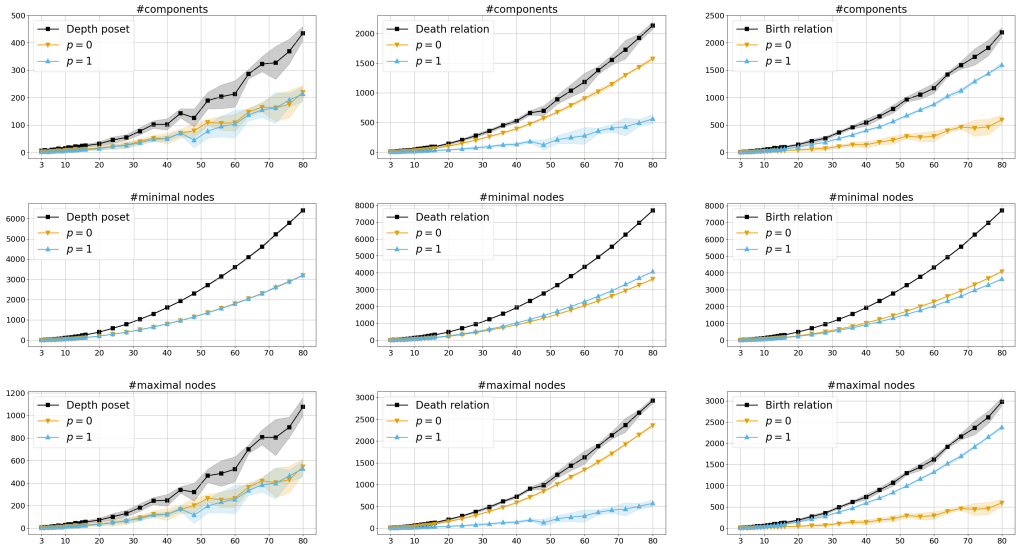


Figure 9 *From left to right:* the average numbers of components, minimal nodes, and maximal nodes of the depth poset, the death relation, and the birth relation of a random function on the 2-torus, respectively. Because of symmetry, the curves for the death and birth relations are nearly identical.

As proved in [9], the depth poset splits by the dimensions of the nodes, but depending on the data it may split further. The upper row in Figure 9 shows that it indeed does so for random functions on the 2-torus. Each component has at least one minimal and one maximal node, but there may be more, and the middle and lower rows of the same figure support this possibility. Recall that the minimal nodes of the depth poset are exactly the shallow pairs, which explains why their number has barely any variance. In $d = 2$ dimensions, there are $4n^2$ incident vertex-edge pairs, and such a pair is shallow iff the edge precedes the other three incident edges of the vertex, and the vertex succeeds the other incident vertex of the edge. Indeed, if an incident vertex-edge pair satisfies these conditions, then it is a pair in persistence as well as shallow, and if it does not satisfy these conditions, it may or may not be a pair in persistence but it is certainly not shallow. Since the vertices and edges are ordered randomly and independently, the edge is first among four with probability $\frac{1}{4}$ and the vertex is second among two with probability $\frac{1}{2}$. Ignoring the extra (-1)-cell, the chance of such a pair to be shallow is therefore $\frac{1}{8}$. Symmetrically, there are $4n^2$ incident edge-square pairs, and ignoring the extra 3-cell, the chance of an incident edge-square pair to be shallow is again $\frac{1}{8}$. This implies that the expected number of shallow pairs is $n^2 + O(1)$; see the first panel in the middle row of Figure 9. Each shallow pair is also a minimal node in the death and birth relations, but both have additional minimal nodes; see the symmetry of the curves in the second and third panels in the middle row of Figure 9, which is a consequence of the symmetry of cubically subdividing the torus. We observe that there are many more minimal nodes than there are maximal nodes. Curiously, the depth poset has many fewer maximal nodes than the death and birth relations, while the difference seems less dramatic for the minimal nodes.

By definition, a poset has no directed cycle, but it may have undirected cycles (pairs of paths that connect the same two nodes). For a 1-dimensional function, the death and birth relations have no undirected cycles either but, perhaps surprisingly, the union of these two relations can have such cycles. Figure 10 shows the average number of (undirected) cycles (which we compute as #components + #arcs − #nodes), together with the average height (defined as the number of arcs in the longest directed path in the relation).

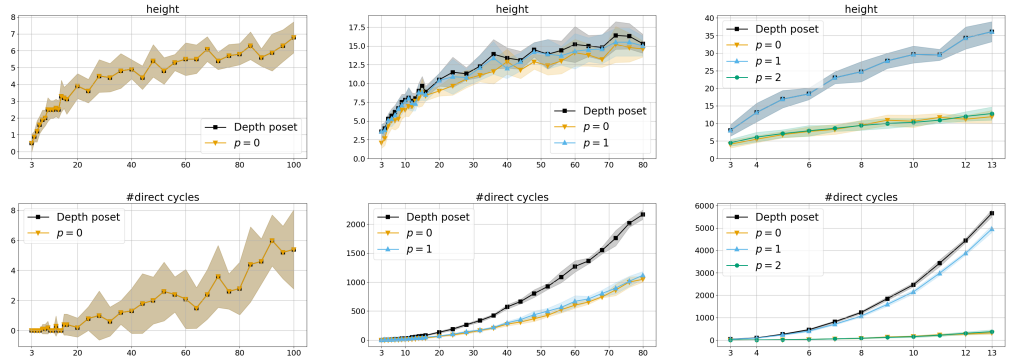


Figure 10 *From left to right:* the average height (*upper row*) and the average number of undirected cycles (*lower row*) in the depth poset of a random function on the d-torus, for $d = 1, 2, 3$, respectively.

4.2 Straight-line Homotopies

Letting $f_0, f_1 \colon K \to \mathbb{R}$ be two random functions on the d-torus, the *straight-line homotopy* between them is the 1-parameter family of functions $f_\lambda(\gamma) = (1 - \lambda)f_0(\gamma) + \lambda f_1(\gamma)$, for all $\gamma \in K$ and $0 \leq \lambda \leq 1$. To collect the statistics, we perform the transpositions of cells in sequence and one at a time. Given two cells of not necessarily the same dimension, $\gamma, \eta \in K$, the parameter at which their values match is

$$\lambda(\gamma, \eta) = \frac{f_0(\eta) - f_0(\gamma)}{[f_0(\eta) - f_0(\gamma)] - [f_1(\eta) - f_1(\gamma)]}. \tag{24}$$

The relevant such parameter values are the ones in $[0, 1]$. Instead of sorting them, it is more efficient to maintain the filter, and at each step transpose the two cells whose values match next. If this pair is not unique, we break ties and transpose the cells in sequence. As shown

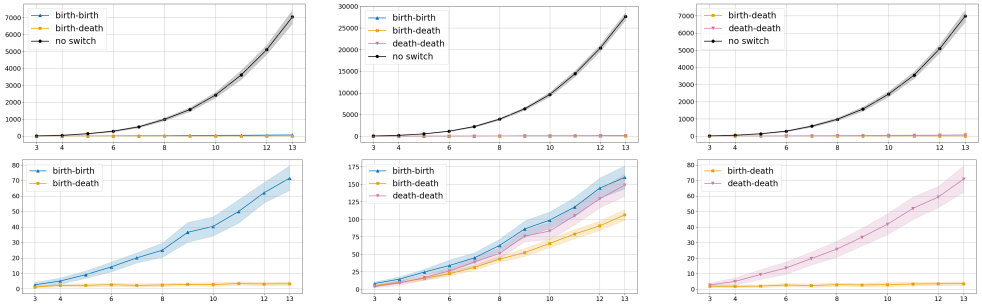


Figure 11 *From left to right:* the average number of transpositions (*upper row*) and switches (*lower row*) between pairs of vertices, edges, and 2-cells for the straight-line homotopy between two random functions on the 2-torus, respectively.

in the upper row of Figure 11, the transpositions that are not switches outnumber those that are switches, so it is useful to draw the curves of the latter on a more appropriate scale, which we do in the lower row of the same figure. As shown in the lower row of Figure 11, there are barely any BD-type switches of two vertices or two 2-cells, and even for edges, there are fewer BD-type switches than switches of BB-type and switches of DD-type.

An interesting aspect of the transpositions is how much they change the depth poset. Every switch affects exactly two nodes, while a transposition that is not a switch leaves all nodes unchanged. The effect on the arcs is less predictable, so we measure the change by counting the arcs that are different before and after the transposition. Not surprisingly, there

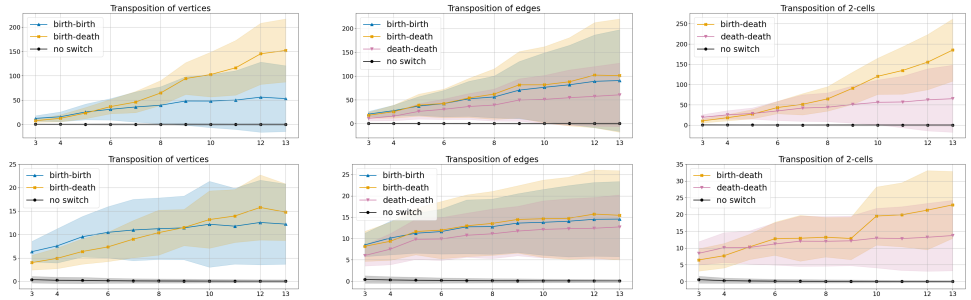


Figure 12 *From left to right*: the average number of arcs in the depth poset (*upper row*) and in the transitive reduction of the depth poset (*lower row*) that are different before and after the transposition of two vertices, two edges, and two 2-cells during a straight-line homotopy between two random functions on the 2-torus, respectively. We distinguish between different kinds of switches and non-switches.

are barely any arcs that change when the transposition is not a switch. Also not surprisingly, there are significantly more arcs that change in the depth poset as compared to the transitive reduction of the poset. Because of the limited size of the random functions in our experiment, it is difficult to gauge how the size of the symmetric difference of the arcs before and after the transposition grows with the size of the function. In any case, it seems to grow much slower than the total number of arcs, which we observe to grow quadratically in n; see the middle panel in the lower row of Figure 8.

5 Discussion

The main contribution of this paper is the complete analysis of how transpositions of cells in the filter affect the depth poset of a filtered Lefschetz complex. This analysis provides further evidence for the relative significance of birth-death switches, as they mark a more extensive re-organization of the persistent homology than all other types of transpositions. Our initial computational experiments suggest that the birth-death switches are more rare than all other types, and that the number of arcs that change grows much slower than the total number of arcs in the depth poset. This work calls for further research in at least the following three directions:

- investigate the links between the transposition induced changes in the depth poset and the bifurcations in the associated gradient dynamics;
- extend the notion of a depth poset from discrete Morse functions to discrete vector fields as introduced in Forman [12], or even to multi-vector fields as studied in [16, 17];
- study the depth poset stochastically for various models of random functions and for homotopies between them.

We finally mention that the algorithms in this paper and its precursor [9] are closely related to the algorithms for topology optimization described by Nigmetov and Morozov [19]. Can the more global topological information provided by the depth poset be harvested to get faster or better methods for topology optimization?

References

1 Dominique Attali, Marc Glisse, Samuel Hornus, Francis Lazarus, and Dmitriy Morozov. Persistence-sensitive simplification of functions on surfaces in linear time. *Manuscript, presented at TOPOINVIS*, 9:23–24, 2009. URL: `https://members.loria.fr/SHornus/simplif/attali-pssfslt.pdf`.

2 Ulrich Bauer. Ripser: efficient computation of Vietoris–Rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423, 2021. doi:10.1007/s41468-021-00071-5.

3 Ulrich Bauer, Carsten Lange, and Max Wardetzky. Optimal topological simplification of discrete functions on surfaces. *Discrete & Computational Geometry*, 47(2):347–377, 2012. doi:10.1007/s00454-011-9350-z.

4 Martin P. Bendsøe. *Optimization of Structural Topology, Shape, and Material*. Springer, Berlin, Germany, 1995. doi:10.1007/978-3-662-03115-5.

5 Martin P. Bendsøe and Ole Sigmund. *Topology Optimization*. Springer, Berlin, Germany, 2nd edition, 2004. doi:10.1007/978-3-662-05086-6.

6 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007. doi:10.1007/s00454-006-1276-5.

7 David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. Vines and vineyards by updating persistence in linear time. In *Proceedings of the 26th Annual Symposium on Computational Geometry*, SCG '06, pages 119–126, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1137856.1137877.

8 Olaf Delgado-Friedrichs, Vanessa Robins, and Adrian Sheppard. Skeletonization and partitioning of digital images using discrete Morse theory. *IEEE Trans Pattern Anal Mach Intell*, 37(3):654–666, March 2015. doi:10.1109/TPAMI.2014.2346172.

9 Hebert Edelsbrunner, Michał Lipiński, Marian Mrozek, and Manuel Soriano-Trigueros. The poset of cancellations induced by gradient dynamics in a filtered Lefschetz complex. arXiv:2311.14364, 2026. doi:10.48550/arXiv.2311.14364.

10 Herbert Edelsbrunner and John L Harer. *Computational topology: an introduction*. American Mathematical Society, Providence, Rhode Island, 2010. doi:10.1090/mbk/069.

11 Herbert Edelsbrunner, Dmitriy Morozov, and Valerio Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, SCG '06, pages 127–134, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1137856.1137878.

12 Robin Forman. Combinatorial vector fields and dynamical systems. *Mathematische Zeitschrift*, 228(4):629–681, 1998. doi:10.1007/PL00004638.

13 Robin Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134(1):90–145, 1998. doi:10.1006/aima.1997.1650.

14 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

15 Solomon Lefschetz. *Algebraic Topology*, volume 27 of *Colloquium Publications*. American Mathematical Society, Providence, Rhode Island, 1942.

16 Michał Lipiński, Jacek Kubica, Marian Mrozek, and Thomas Wanner. Conley-Morse-Forman theory for generalized combinatorial multivector fields on finite topological spaces. *Journal of Applied and Computational Topology*, 7:139–184, 2022. doi:10.1007/s41468-022-00102-9.

17 Marian Mrozek. Conley–Morse–Forman theory for combinatorial multivector fields on Lefschetz complexes. *Foundations of Computational Mathematics*, 17(6):1585–1633, 2017. doi:10.1007/s10208-016-9330-z.

18 Marian Mrozek and Thomas Wanner. *Connection Matrices in Combinatorial Topological Dynamics*. SpringerBriefs in Mathematics. Springer Cham, 2025. doi:10.1007/978-3-031-87600-4.

19 Arnur Nigmetov and Dmitriy Morozov. Topological optimization with big steps. *Discrete & Computational Geometry*, 72(1):310–344, 2024. doi:10.1007/s00454-023-00613-x.

20 Jens Kristian Refsgaard Schou and Bei Wang. Persisort: A new perspective on adaptive sorting based on persistence. In *Proc. 36th Canadian Conference on Computational Geometry*, pages 287–302, July 2024.

Compressed Data Structures for Heegaard Splitting

Henrique Ennes
Université Côte d'Azur, Nice, France
INRIA d'Université Côte d'Azur, Sophia Antipolis, France

Clément Maria
INRIA d'Université Côte d'Azur, Sophia Antipolis, France

Abstract

Heegaard splittings provide a natural representation of closed 3-manifolds by gluing handlebodies along a common surface. These splittings can be equivalently given by two finite sets of meridians lying on the surface, which define a Heegaard diagram. We present a data structure to effectively represent Heegaard diagrams as normal curves with respect to triangulations of a surface of complexity measured by the space required to express the normal coordinates' vectors in binary. This structure can be significantly more compressed than triangulations of 3-manifolds, giving exponential gains for some families. Even with this succinct definition of complexity, we establish polynomial-time algorithms for comparing and manipulating diagrams, performing stabilizations, detecting trivial stabilizations and reductions, and computing topological invariants of the underlying manifolds, such as their fundamental and homology groups. We also contrast early implementations of our techniques with standard software programs for 3-manifolds, achieving faster algorithms for the average cases and exponential gains in speed for some particular presentations of the inputs.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Geometric topology; Theory of computation → Data structures design and analysis

Keywords and phrases 3-manifold, Heegaard splitting, curves on surfaces, surface theory, data structure, computational topology

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.42

Related Version *Full Version*: `https://arxiv.org/abs/2507.11406`

Supplementary Material *Software (Source Code)*: `https://doi.org/10.5281/zenodo.19236220`

Funding This work has been partially supported by the ANR project ANR-20-CE48-0007 (AlgoKnot) and the project ANR-15-IDEX-0001 (UCA JEDI).
Henrique Ennes: The PhD thesis, which this work is a part of, has been supported by the French government, through the France 2030 investment plan managed by the Agence Nationale de la Recherche, as part of the "UCA DS4H" project, reference ANR-17-EURE-0004.

1 Introduction

Since the early days of computational topology, *3-manifolds* have attracted researchers' keen interest [27, 31, 40] for lying exactly midway between the well-understood surfaces and the wildly behaved 4-manifolds. For example, several 3-manifold problems were related to important questions in complexity theory: S^3 recognition was proved by Schleimer to be in NP [50], whereas approximations of some quantum invariants are firmly tied to quantum computing and are known to be #P-hard [1, 2]. Similarly, the problems of classifying [17, 39] and manipulating the geometry and topology of these spaces [8, 42] have also received significant theoretical and practical attention, with implementations of different methods publicly available in the most important software in the field, `Regina` [12] and `SnapPy` [18].

The algorithmic point of view is particularly relevant for the construction of *censuses* of homomorphism classes of 3-manifolds [7, 9, 10, 13]. Drawing inspiration from the related problem of knot classification (this time, up to ambient isotopies), where the cumulative work

© Henrique Ennes and Clément Maria;
licensed under Creative Commons License CC-BY 4.0

of researchers [11, 15, 21] has culminated in the widely accessible census of $\sim 350'000'000$ prime knot diagrams, computational methods are expected to foster the elaboration of tables of 3-manifolds, particularly valuable assets in the proposition and verification of conjectures, as well as in the discovery of counterexamples.

Abstracting modern approaches [6, 9, 11, 13] of the construction of censuses of topological spaces, one requires (1) compact data structures to represent massive amounts of (candidate) spaces and the iterative construction of all spaces up to a maximum complexity; (2) efficient operations to disqualify inputs (such as composite knots, links, non-prime 3-manifolds) and simplify the representation combinatorics (Reidemeister moves, Pachner moves, etc); and, (3) methods for discriminating spaces, such as invariants. For 3-manifolds, these steps are usually addressed through *triangulations* – homeomorphic copies described by a finite set of tetrahedra and gluing rules among their faces – of complexity often measured by the number of tetrahedra. On the other hand, theoretical bounds on the minimal number of tetrahedra to represent some spaces [19, 33, 34, 44] and the difficulty of generating particular structures, such as hyperbolicity [44], can limit the effectiveness of triangulation-based censuses.

Heegaard splittings emerge as an alternative to triangulations. They depict a closed 3-manifold M as the result of gluing two better-behaved pieces called *handlebodies* along a common boundary. Splittings can also be combinatorially described by *Heegaard diagrams*, two sets of curves in a surface whose intersection pattern conveys a lot of topological information about M. We suggest here a new data structure to represent Heegaard diagrams based on *normal coordinates*, of a size as large as *binary* of the number of intersections between the two sets of curves. The gain in measuring the complexity of 3-manifolds through this notion is not to be overlooked: we present in Section 5 families of manifolds whose diagrams' complexity grows linearly with the size of the input, but whose triangulations necessarily use exponentially many tetrahedra. However, what is more important is that due to the proliferation of efficient operations in normal coordinates [24, 41, 48, 49, 53], we can establish polynomial-time algorithms even in this compressed notion of complexity, leading to basic manipulations of Heegaard splittings and the computation of simple 3-manifold invariants such as the fundamental and homology groups. Moreover, as we point out in Remark 9, some important complexity results about 3-manifolds – such as sphere (co-)recognition – are still preserved if we assume compressed inputs, which tells us much about the inherent difficulty of these problems. We hope that these techniques are useful for the study and classification of topological spaces that are beyond the reach of efficiency when represented by other methods.

The paper begins with a background section on both low-dimensional and computational topology (Section 2), where we review some of the concepts and results that will be needed in the later parts. Our contribution starts in Section 3, where we formally present our data structure for Heegaard splittings. Section 4 establishes basic algorithms in this structure, which are further expanded the paper's full version. We compare early implementations of these algorithms with triangulation-dependent methods in Section 5.

Before we close this introduction, we draw the reader's attention to two subtleties of the current work. First, although most of our tools apply to *generalized* Heegaard splittings defined for *orientable compact* 3-manifolds, potentially with boundary [37, 51], we will focus solely on *closed* 3-manifolds. Therefore, all 3-manifolds here will be assumed orientable and closed, unless explicitly stated otherwise. Second, we assume throughout a *unit-cost RAM model* with some bit-size W [26]. This means that the four integer operations and integer comparison can be executed in constant time, provided that the numbers involved have binary representations of size at most W.

2 Background material

2.1 Background on low-dimensional topology

We assume the reader is acquainted with the basic notions of surface theory at the level of [25], referring to the paper's full version for details. We use Σ to denote a general surface and Σ_g the (unique) closed surface of genus g. Whenever possible, we reserve Greek letters for multicurves and Latin letters for curves and arcs. All (multi)curves are assumed simple. As usual, we do not differentiate (multi)curves from their isotopy classes if no confusion arises.

The *mapping class group* of the surface Σ_g, $\mathrm{Mod}(\Sigma_g)$, is the group of orientation-preserving $\Sigma_g \to \Sigma_g$ homeomorphisms up to isotopy. For each g, $\mathrm{Mod}(\Sigma_g)$ is finitely generated as a group by an alphabet $\mathcal{L}$, consisting of *Dehn twists* about $3g-1$ (isotopy classes of) canonically defined essential curves $s_1, \ldots, s_{3g-1}$ [43]. Formally, a Dehn twist $\tau_s : \Sigma_g \to \Sigma_g$ about s is given by cutting out a cylindrical regular neighborhood around the curve, applying a 2π left-twist to one of the cylinder's boundaries, and gluing the cylinder back into the surface. We call the curves $s_1, \ldots, s_{3g-1}$ the surface's *Lickorish generators*. Hopefully without confusion, we also call the Dehn twists in the alphabet $\mathcal{L} = \{\tau_{s_1}, \ldots, \tau_{s_{3g-1}}\}$ Lickorish generators.

An essential multicurve γ in Σ_g is a *system* if Σ_g cut along γ, $\Sigma_g \backslash \gamma$, is a $2g$ punctured sphere. A system γ in Σ_g can be used to construct the genus g *handlebody* $\mathcal{H}_\gamma$ where we attach 2-handles along the curves γ in $\Sigma \times \{0\}$ and fill any spherical boundary component with 3-handles. Note that $\partial \mathcal{H}_\gamma = \Sigma_g$ and that $\mathcal{H}_\gamma$ is homeomorphic to the direct sum of g solid tori. Components of γ are *meridians* of $\mathcal{H}_\gamma$ in the sense that they bound disks in the handlebody's interior. Two systems γ and γ' in Σ_g will be called *equivalent* if γ' is a set of meridians in $\mathcal{H}_\gamma$ (which implies that γ is a set of meridians in $\mathcal{H}_{\gamma'}$).

Let c_1, c_2 be two components of a system γ and p an arc from a puncture corresponding to c_1 to a puncture corresponding to c_2 in $\Sigma_g \backslash \gamma$. The *band sum* of c_1, c_2 and p is the curve c in $\Sigma_g \backslash \gamma$ obtained by the surgery of Figure 1. One can show that c is a meridian in $\mathcal{H}_\gamma$ [22, 37], so $\gamma' = \gamma - \{c_1\} \cup \{c\}$ is a system equivalent to γ. We call this whole operation a *disk slide*. With finitely many applications of disk slides and isotopies, a system γ can be transformed into any equivalent system γ' [37].

A genus g *Heegaard splitting* is the closed 3-manifold resulting from the gluing of two handlebodies with boundaries homeomorphic to Σ_g through a gluing map $f : \Sigma_g \to \Sigma_g$. In fact, one might assume that f is given by a $\phi \in \mathrm{Mod}(\Sigma_g)$ composed with some canonical orientation-reversing homeomorphism. This means that a Heegaard splitting can be represented by a pair (Σ_g, ϕ) where ϕ is a word on Lickorish generators. We call this pair the *Heegaard word* for the splitting. Every closed 3-manifold has a Heegaard splitting [46, 47].

Note that if we fix a system of meridians α on one of the two handlebodies, $\mathcal{H}_\alpha$, then $\beta = \phi(\alpha)$ is a system of meridians of the *other* handlebody, $\mathcal{H}_\beta$. We call the tuple $(\Sigma_g, \alpha, \beta)$ a *Heegaard diagram*. Heegaard diagrams are algorithmically independent of words: any two systems α and β will define the same splitting, provided that the curves are meridians in their respective handlebodies. Diagrams are therefore not unique for a given splitting. For example, if β and β' are *isotopic diagrams* on Σ_g – that is, β and β' are isotopic multicurves in Σ_g – then $(\Sigma_g, \alpha, \beta)$ and $(\Sigma_g, \alpha, \beta')$ define the same manifold. Similarly, if β is equivalent to β', we say that the diagrams are *equivalent* and $(\Sigma_g, \alpha, \beta)$ and $(\Sigma_g, \alpha, \beta')$ also define the same 3-manifold. In this sense, disk slides play for Heegaard diagrams a role similar to the Reidemeister moves for knots.

A splitting $(\Sigma_g, \mathcal{H}_\alpha, \mathcal{H}_\beta)$ is *reducible* if there is a curve c in Σ_g that is a meridian of both $\mathcal{H}_\alpha$ and $\mathcal{H}_\beta$. We call a diagram $(\Sigma_g, \alpha, \beta)$ *reducible* if either (1) there is a component $a \in \alpha$ and a component $b \in \beta$ that are isotopic in Σ_g; or (2) there is a separating curve s, disjoint

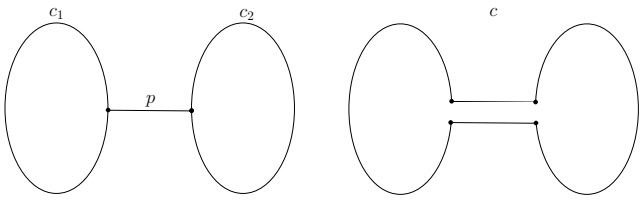


Figure 1 The result c of a band sum of c_1, c_2, and p.

from both α and β, and such that $\Sigma_g \backslash s$ defines a connected sum of two Heegaard splittings of smaller genera. The uniqueness of fundamental group decomposition [3] and [51][Theorem 6.3.5] imply that a reducible diagram represents a reducible splitting. A reducible splitting can always be written as the connected sum of sub-splittings [3, 37].

A presentation $\langle x_1, \ldots, x_g | r_1, \ldots, r_g \rangle$ of the *fundamental group* $\pi_1(M)$ can be obtained as follows [30, 37]: label the curves $a_1, \ldots, a_g \in \alpha$ and $b_1, \ldots, b_g \in \beta$ and give them an orientation; then traverse each curve b_i along the assumed orientation and, whenever it intersects some a_j, we append x_j to the relation r_i if the index of the intersection is positive, otherwise we append x_j^{-1} to it. Because the *first (integral) homology group of* M, $H_1(M)$ is the abelianization of $\pi_1(M)$ [28], we can present $H_1(M)$ as $\langle x_1, \ldots, x_g | r'_1, \ldots, r'_g, [x_i, x_j] \rangle$, where each $r'_i = x_1^{k_i^1} \ldots x_g^{k_i^g}$ can be computed by commuting the generators x_j in the relations r_i. Note that k_i^j equals the algebraic intersection number $\hat{i}(b_i, a_j)$ and we store this information in a $g \times g$ *presentation matrix* for $H_1(M)$, whose Smith normal form carries all the necessary information of the homology group.

2.2 Background on computational topology

Heegaard diagrams consist of three essential components that any candidate data structure must encode: a surface, Σ_g, and two multicurves, α and β. Starting with the surface, a graph $T = (V, E)$ embedded in Σ is a *cellular embedding* if the manifold defined by cutting Σ along T, $\Sigma \backslash T$, is homeomorphic to a collection of disks. We call each component of $\Sigma \backslash T$ a *face* of T. A cellular embedding T in the surface Σ is said to be *oriented* if we assign to each face an ordered list of its vertices, consistent with the orientation of Σ as in the right-hand rule. We assume every cellular embedding to be oriented. A cellular embedding T is a *(generalized) triangulation* of Σ if exactly three edges of T bound each face. We define the size of a triangulation, $|T|$, as the number of faces; note that $|V|, |E| = O(|T|)$ where $|V|$ is the number of vertices of the triangulation and $|E|$ is the number of edges. Surface triangulations are not to be confused with the similar concept of 3-manifold triangulations.

Every oriented surface can be described as a data structure by a triangulation. For the triangulations themselves, we assume that they are given as a set of vertices, edges, and triangles, with bidirectional pointers between vertices and edges, and between edges and the triangles, therefore requiring a total space of $O(|T|)$. In particular, determining whether two triangles are adjacent, getting the incident vertices and edges of a triangle, and finding triangles bounded by a specific edge take constant time. Other, more compact data structures for triangulation, such as CGAL's [20], can be converted into our structure in time $O(|T|)$ and at a similar cost to memory. More general cellular embeddings are treated similarly.

A curve c is *edged* with respect to T if it is disjoint from its faces. We represent an edged curve as a data structure by the list $E_T(c) = \{e \in E : |c \cap e| = e\}$, assumed to be ordered according to an implicit traversing orientation of c. Similarly, a multicurve γ can be represented by a set of $\#\gamma$ edge lists, one for each component, which we denote as $E_T(\gamma)$. We

assume that no two components can share an edge, nor can the same component be incident to a vertex more than once; nevertheless, two distinct components can share a vertex. The *complexity* of $E_T(\gamma)$ is $\|E_T(\gamma)\| = \sum_{c\in\gamma} |E_T(c)|$.

On the other hand, a multicurve γ is called *standard* in T if it intersects the graph only at the edges and transversely. If γ is standard, we represent it as a collection of words in a finite alphabet: arbitrarily orient the edges E and take $E^{\pm} = E \cup E^{-1}$ for the letters. Then for each component c of γ, the word $I_T(c)$ in $(E^{\pm})^*$ is given by traversing c along some arbitrary direction and, whenever we intersect an edge $e \in E$, we append e^i to $I_T(c)$ where i is the index of the intersection. An *intersection sequence* representation for γ, $I_T(\gamma)$, is the set $\{I_T(c)\}_{c\in\gamma}$. While $E_T(\gamma)$ is well defined at the multicurve level, intersection sequences are defined only up to isotopies inside the cellular embedding's faces. Moreover, cyclic permutations and taking inverses of the words $I_T(c)$ in $I_T(\gamma)$ define the same isotopy classes.

Instead of working with explicit intersection sequences, we will often assume *straight line programs* (*SLPs*) to express $I_T(c)$. Formally, an SLP on the alphabet $E^{\pm}$ is a finite sequence of equations $[x_1 = \mathrm{EXPR}_1, \ldots, x_\ell = \mathrm{EXPR}_\ell]$, where the left-hand side, x_i, are symbols not in $E^{\pm}$ called *variables*, and the expressions on the right-hand side, $\mathrm{EXPR}_i \in (E^{\pm} \cup \{x_1, \ldots, x_{i-1}\})^*$, are called *assignments*. Each assignment EXPR_i is assumed to be a symbol from $E^{\pm}$, in which case the assignment is called *simple*, or of the form $\mathrm{EXPR}_i = x_{j_1}^{\pm 1} \cdot x_{j_2}^{\pm 1} \cdot \cdots \cdot x_{j_k}^{\pm 1}$, where $1 \le j_1, \ldots, j_k < i$, in which case we say the assignment is *proper*. In particular, the SLP $[x_1, \ldots, x_\ell]$ represents a word w in $(E^{\pm})^*$ if recursively substituting the appropriate x_j in each proper assignment, gives x_ℓ equal to w. We call ℓ the *length* of the SLP and define its *complexity* $m = \sum_{i=1}^{\ell} |\mathrm{EXPR}_i|$, where $|\mathrm{EXPR}_i|$ is the number of variables and symbols in the assignment $|\mathrm{EXPR}_i|$. Given any word $e_1 \ldots e_n$ in $(E^{\pm})^*$, one can trivially build an SLP of complexity $2n$, but there are instances in which SLPs can be used to give exponential gains over uncompressed representation of words [49]. Nonetheless, even in extremely compressed cases, several useful operations can still be efficiently performed, as stated in the following lemma, proved in the paper's full version. Here and throughout, we assume that $\lg k$ is the number of bits necessary to represent the potentially signed integer k; in particular, $\lg k = \log k + 2$ if k is signed and $\log k + 1$ if unsigned.

▶ **Lemma 1.** *Let $\langle x_1, \ldots, x_\ell \rangle$ be an SLP of complexity m representing a word $w \in (E^{\pm})^*$. In a unit-cost RAM model of bit-size $m + \lg k$, for a fixed $k \in \mathbb{Z}^+$, one can compute*

(a) *an SLP of w^{-1} in time $O(1)$;*

(b) *an SLP of w^k in time $O(\lg k)$;*

(c) *the number of occurrences of a symbol $e \in E^{\pm}$ in w, $e(w)$, in time $O(m)$.*

We will assume that every intersection sequence $I_T(c)$ is given by an SLP of complexity $\|I_T(c)\|$ and let $\|I_T(\gamma)\| = \sum_{c\in\gamma} \|I_T(c)\|$.

Now, suppose that $T = (V, E)$ is a triangulation. A standard (multi)curve γ is *normal* if it intersects each face of T in *elementary arcs* that connect distinct sides of the triangle in which they are contained. Equivalently, γ is normal if each (uncompressed) intersection sequence in $I_T(\gamma)$ is *cyclic reduced*, that is, if there are no substrings of type ee^{-1} or $e^{-1}e$ in any of the cyclic permutations of the components' sequences. When the curve c is normal for a triangulation, its isotopy class is fixed by the number of times it intersects each labeled edge [48], implying that we can describe it through a vector in $N_T(c)$ in $\mathbb{N}^{|E|}$. We call $N_T(c)$ the curve's *normal coordinates*, whose complexity, $\|N_T(c)\|$, is the number of bits necessary to store this vector, i.e., $\sum_{e\in E} \lg e(c) = \sum_{e\in E} (\log e(c) + 1)$. Naturally, we define $N_T(\gamma)$ as the set $\{N_T(c)\}_{c\in\gamma}$.

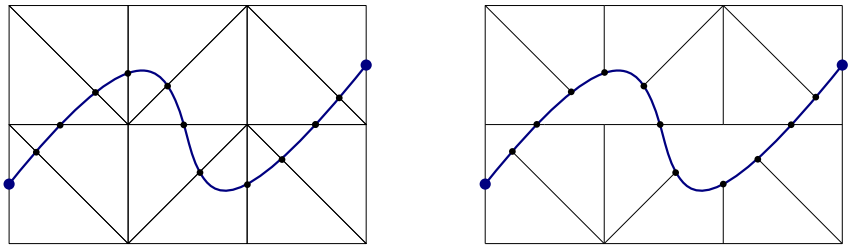


Figure 2 Left: the overlay graph of a normal arc (blue) with respect to a triangulation (black). Right: its street complex.

▶ Remark 2. We could represent a normal multicurve γ with a *single* vector $\sum_{c\in\gamma} N_T(\gamma)$. Similarly, it is possible (and common) to extend the notion of SLPs to encode more than one uncompressed word, which, in our case, implies giving all components of γ as a single SLP. Although these approaches are more parsimonious, they are less useful for our needs. Moreover, Lemma 5.1 of [24] gives an algorithm to transform, in time $O(|T|\|N_T(\gamma)\|)$, the normal coordinates of a multicurve given by a sum of its components' coordinates to our expected form.

Any normal (multi)curve γ partitions the edges of T into segments called *ports*. The *overlay graph* of γ is defined by having as vertices the union of V and the intersections $E \cap \gamma$, and as edges the union of the ports and the elementary arcs of γ (see Figure 2). The overlay graph splits each triangle t of the original triangulation into two types of regions: *junctions*, which are adjacent to three edges of T, and *blocks*, which are adjacent to only two. A port of the overlay graph is called *redundant* if it separates blocks. Following Erickson and Nayyeri [24], we define the *street complex* of γ, $S(T,\gamma)$, by deleting redundant ports and their endpoints from the overlay graph. In practice, the street complex merges blocks into longer faces called *streets*; naturally, $S(T,\gamma)$ is a cellular embedding in the surface triangulated by T, see Figure 2. We define the *crossing sequence* of a street in $S(T,\gamma)$ as the intersection sequence of a curve that crosses the street from one end to the other with respect to the original triangulation T. For simplicity, we assume that any port that separates two junctions is a degenerate street with a crossing sequence of length 1. The street complexes of all multicurves in this paper will have complexity $O(|T|)$ [24].

3 Data structure for Heegaard diagrams

Normal coordinates suggest a very natural encoding of Heegaard diagrams and, consequently, of closed 3-manifolds. Explicitly, we propose representing $(\Sigma_g, \alpha, \beta)$ by a tuple $(T, E_T(\alpha), N_T(\beta))$, where $T = (V, E)$ is a triangulation, $E_T(\alpha)$ is an edge representation of α, and $N_T(\beta)$ is a normal coordinate representation of β. We will often refer to the pair $(T, E_T(\alpha))$ as a *marked triangulation* of Σ_g, in which context $N_T(\beta)$ will be called the diagram itself. The *complexity* of the diagram is

$$\|(T, E_T(\alpha), N_T(\beta))\| = |T| + \|E_T(\alpha)\| + \|N_T(\beta)\|. \tag{1}$$

We note that $\|E_T(\alpha)\| = O(|T|) = O(\|N_T(\beta)\|/|\beta \cap E|)$, that is, for complexity purposes, only the last term of (1) matters. However, for the sake of clarity, we will still refer to the terms of equation (1) arising from the marked triangulation in our analyses. All complexities will be reported under the assumption that $|\beta \cap T| = \Omega(|T|^2)$, as otherwise there are some faster trivial algorithms.

Our choices of data structures for α and β allow us to represent curves of exponentially many intersection points with respect to the complexity of the diagram. Referring to the discussion at the end of Section 2.1, more intersections may lead to more complicated fundamental groups and, consequently [44][Proposition 2.6.6], more complicated 3-manifolds. In this sense, our structure is significantly more compressed than representations of Heegaard diagrams with *both* α and β edged, which can have, at most, $E_T(\alpha) \times E_T(\beta) = O(|T|^2)$ intersections.

Complexity gains are also noticeable if we compare our structure with triangulations of 3-manifolds. The usual algorithm to triangulate Heegaard diagrams [29] takes polynomial-time in $|\alpha \cap \beta|$. Moreover, as we empirically verify in Section 5, converting a Heegaard word to a diagram can be exponentially faster than converting the word to a triangulation for some choices of input and at least as fast for the general case. On the downside, it is unclear how to obtain and manipulate geometric information (such as *hyperbolicity*) or to compute certain quantum invariants directly from Heegaard diagrams, making triangulations still preferable for these kinds of problems.Refer to Section 6 of the paper's full version.

As stated in the introduction, our goal is to establish basic efficient algorithms even in this very compressed setting. For such a purpose, we will make extensive use of the next two results. The first of them is (almost) fully established by Erickson and Nayyeri [23].

▶ **Theorem 3.** *Suppose that $(T, E_T(\alpha), N_T(\beta))$ is a genus g Heegaard diagram of complexity m. In a unit-cost RAM model of bit-size $\log|\beta \cap E| = O(m/|T|)$, one may compute*

(a) *the street complex $S(T, \beta)$ in time $O(m|T|)$;*

(b) *SLPs of total complexity $O(m|T|)$ of the crossing sequence of* all *streets of $S(T, \gamma)$ in time $O(m|T|)$;*

(c) *SLPs $I_T(\beta)$ of complexity $O(m)$, in time $O(m)$.*

For the next theorem, we might assume that a Heegaard word ϕ is given in a *power-notation form*, that is, $\phi = \tau_{s_n}^{k_n} \circ \cdots \circ \tau_{s_1}^{k_1}$, $s_i \neq s_{i+1}$ for $1 \leq i < n$, and with each $k_i \in \mathbb{Z}$ in signed binary. A *fully uncompressed* Heegaard word $\phi' = \tau_{s'_{|\phi'|}}^{\pm 1} \circ \ldots \tau_{s'_1}^{\pm 1}$, with some s'_i potentially equal to s'_{i+1}, can be put in power-notation form in time $O(|\phi'|)$ with a unit-cost RAM model of bit-size $\lg|\phi'|$. For convenience, we shall also assume that the Lickorish generators are given as edged curves with respect to T. Nevertheless, even if we suppose that $\mathcal{L}$ is given by normal coordinates or intersection sequences, the theorem still holds with similar complexity bounds by multiple applications of Theorem 4.9 of the full version.

▶ **Theorem 4.** *Suppose $\phi = \tau_{s_n}^{k_n} \circ \cdots \circ \tau_{s_1}^{k_1}$, $s_i \neq s_{i+1}$ for $1 \leq i < n$, is a Heegaard word in $\mathrm{Mod}(\Sigma_g)$ and let T be a triangulation of Σ_g with edged α and Lickorish generators. There is an algorithm to compute intersection sequences $I_T(\phi(\alpha))$ in time $\Theta(g|T| + n|T|\lg k)$ in a unit-cost RAM model of bit-size $\lg k$ where $\lg k = \sum_{i=1}^{n} \lg k_i$. Moreover, the Heegaard diagram $(T, E_T(\alpha), N_T(\phi(\alpha)))$ of complexity $O(|T|^3 \lg^2 k)$ can be computed in time $O((|T| \lg k)^{10})$.*

Proof. For each $s \in \mathcal{L}$, denote its edge list by $E_T(s) = \{e_{s,1}, \ldots, e_{s,\|E_T(s)\|}\}$, where $e_{s,j} \in E$ for $1 \leq j \leq \|E_T(s)\|$, and define $I_T(s, j)$ as (an SLP of) the intersection sequence of a normal curve isotopic to s starting at a face adjacent to $e_{s,j}$. We can compute $I_T(s, j)$ for each $1 \leq j \leq \|E_T(s)\|$ in time $O(g|T|)$ (Lemma 3.4 in the full version). We also compute $I_T(a)$ for each $a \in \alpha$; in total, this initial step takes time $O(g|T|)$.

Let $I_T(\beta_0) = I_T(\alpha)$. Assuming $k_i > 0$, we recursively construct $I_T(\beta_i)$, for $1 \leq i \leq n$, by replacing any occurrence of an edge $e_{s_i,j} \in E_T(s_i)$ in the SLPs $I_T(\beta_{i-1}) = \{[x_1 = \mathrm{EXPR}_1, \ldots, x_p = \mathrm{EXPR}_p]\}_{b \in \beta}$ with an SLP $[y_1, \ldots, y_q]$ of $I_T(s_i)^{k_i} \cdot s_{i,j}$, that is,

$$I_T(\beta_i) = \{[y_1, \ldots, y_q, x'_{q+1} = \mathrm{EXPR}'_1, \ldots, x'_{q+p} = \mathrm{EXPR}'_q]\}_{b \in \beta}$$

where EXPR'_r is the word EXPR_r with each occurrence of x_t substituted by x'_{p+t} and each occurrence of $s_{i,j}$ substituted by y_q. The case of $k_i < 0$ is treated similarly. By Lemma 1, we note that this can be done in time $O(\|I_T(\beta_{i-1})\| + \lg k_i \|I_T(s_i)\|)$ in the assumed model of computation. In particular, $I_T(\beta_i)$ is an intersection sequence representation of $\beta_i = \tau_{s_i}^{k_i} \circ \cdots \circ \tau_{s_1}^{k_1}(\alpha)$ and has complexity

$$\begin{aligned} O(\|I_T(\beta_i)\|) &= O(\|I_T(\beta_{i-1})\| + \lg k_i \|I_T(s_i)\|) \\ &= O\left(\|I_T(\alpha)\| + \sum_{i'=1}^{i} \lg k_{i'} \|I_T(s_{i'})\|\right) \\ &\leq O\left(|T| + |T| \sum_{i'=1}^{i} \lg k_{i'}\right) = O\left(|T| \sum_{i'=1}^{i} \lg k_{i'}\right), \end{aligned}$$

where in the second line we recursively applied the equation of the first. The total time to compute $I_T(\beta) = I_T(\beta_n)$ is bounded by $n \times \|I_T(\beta)\| = O(n|T| \lg k)$, and, in the case $k_i = 1$ for all $1 \leq i \leq n$, this becomes $\Theta(n^2|T|) = \Theta(n|T| \lg k)$.

Although $I_T(\beta)$ is, by construction, standard in T, it might not be normal. We can use the main algorithm of [45] to deterministically compute, in time $O(\|I_T(\beta)\|^{10})$, cyclic reduced SLPs for β with complexity $O(\|I_T(\beta)\|^2)$. Alternatively, this can be done with a randomized algorithm in time $O(\|I_T(\beta)\|^3)$ with probability of error at most δ and in a unit-cost RAM model of bit-size $O(\|I_T(\beta)\| + \lceil 1/\delta \rceil)$ [53]. Part (c) of Lemma 1 finishes the proof. ◀

4 Algorithms on Heegaard diagrams

We now describe some polynomial-time algorithms and operations on a genus g Heegaard diagram $(T, E_T(\alpha), N_T(\beta))$ with complexity m and representing a closed 3-manifold M. We choose here to highlight algorithms that refer back to the discussion on censuses from the introduction, giving examples on how to simplify the underlying combinatorics of splittings (Theorem 5), disqualify inputs (Theorem 6), and compute invariants (Theorems 7 and 8). For the complexities reported, we assume a unit-cost RAM model of bit-size $\log |\beta \cap E| = O(\|N_T(\beta)\|/|T|)$.

▶ **Theorem 5** (`DO A DISK SLIDE`). *A diagram $(T, E_T(\alpha), N_T(\beta'))$, where β' is the result of applying a disk slide on any two components of β, can be computed in time $O(m|T|)$.*

Proof. Compute the street complex $S(T, \beta)$; recall that it has complexity of $O(|T|)$. Using breadth-first in the dual graph of $S(T, \beta)$, find, in time $O(|T|^2)$, the shortest arc p that connects faces adjacent to b_1 and b_2 and that does not cross any component of β, which can be avoided by adding infinite weights to the edges of $S(T, \beta)$ not in T. Note that such an arc must exist because no component of β can be separating in the surface. Moreover, because p is shortest and therefore intersects at most $O(|T|)$ streets in $S(T, \beta)$, we can compute using Part (b) of Theorem 3, an SLP of complexity $O(m|T| + |T|) = O(m|T|)$ of an intersection sequence of p with respect to T, call it $I_T(p)$.

Suppose that p is not trivial, that is, it is not just a single point. By Theorem 6.2 of [24], find, in time $O(m|T|)$, SLPs of the intersection sequences of b_1 and b_2 with respect to T starting at edges adjacent to the endpoint faces of p, call them $I_T(b_1)$ and $I_T(b_2)$. Finally, let $I_T(b) = I_T(b_2)^{-1} \circ I_T(p)^{-1} \circ I_T(b_1) \circ I_T(p)$. Note that $I_T(b)$ already is an intersection sequence of a normal curve in T. For, since p is a shortest path, it is normal in $S(T, \beta)$, which implies that it is also normal in T. Moreover, p has junctions as endpoints; if it had streets as endpoints, there would be a shorter arc connecting b_1 to b_2 starting at one of

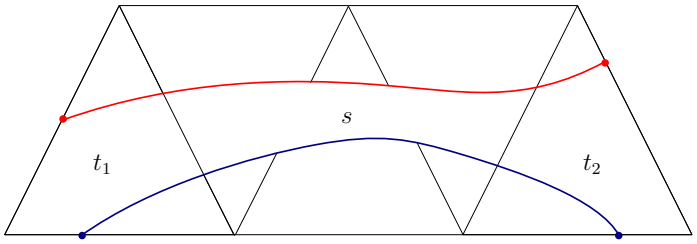


Figure 3 Diagram for the proof of Theorem 5. The red and blue curves represent b_1 and b_2, respectively. The black lines represent the other edges of $S(T, \beta)$. The edges e_1 and e_2 are adjacent to s and t_1 and t_2, respectively.

the ends of the street. Therefore, any other reduction in $I_T(b)$ would necessarily arise by p entering a junction of $S(T, \beta)$ through a non-redundant port e that is also intersected by either b_1 or b_2. But then, the other triangle in T adjacent to e is adjacent to either b_1 or b_2, implying that p is not shortest. By repeated applications of Part (c) of Lemma 1, one can compute, in time $O(|T| \times m)$, the number of (unsigned) occurrences of each edge of E in the SLP of $I_T(b)$, resulting in $N_T(b)$.

If p is trivial, then b_1 and b_2 bound the same face in $S(T, \beta)$. Nonetheless, because the curves are not isotopic, there exists at least one face bounded by b_1 and not bounded by b_2. We call such a face *good*. Choose, in time $O(|T|)$, a non-good face t_1 that is adjacent to a good face through an edge e_1, see Figure 3. Note that e_1 is the endpoint of a street s of sides in b_1 and b_2. Look for the other end of this street, e_2 – again, because b_1 and b_2 are not isotopic, the street has another end. Let t_2 be the face (in fact, a junction) not equal to s, but also adjacent to e_2. Compute, again through [24][Theorem 6.2], the sequence of faces adjacent to b_1 starting at t_1 and going all the way up to t_2 without crossing s, call this sequence $I_T(\widetilde{b_1})$. Compute the related sequence for b_2, this time starting in t_2, call it $I_T(\widetilde{b_2})$. Let $I_T(b) = I_T(\widetilde{b_2}) \circ I_T(\widetilde{b_1})$, note that it is also normal in T as b_1 and b_2 are normal and it enters t_1 and t_2 through distinct edges. Part (c) of Lemma 1 finishes this case. ◀

▶ **Theorem 6** (`DETECT REDUCTION`). *There is an algorithm that runs in time $O(\text{poly}(m, |T|))$ and detects whether a diagram is reducible. Moreover, if the diagram is reducible, it returns either an isotopic pair of $a \in \alpha$ and $b \in \beta$, or a separating curve s.*

Proof. We divide the algorithm into two parts, corresponding to the two hypotheses of reducibility of diagrams, namely the existence of an isotopic pair of meridians $a \in \alpha$ and $b \in \beta$, or of an essential separating curve that splits Σ_g into subdiagrams. For the first case, apply [41][Theorem 6.3] to find, in polynomial-time on the input, an isotopic diagram (α, β') in efficient position. This procedure increases the complexity of β to $m' = m(1 + 2|T|)$ We note that although this subroutine is polynomial-time on the input, the author of [41] does not provide an estimation of the polynomial degree, and any attempt on our part to give one would be out of the scope of this paper. Because we made the diagram efficient, there can only be a component $a \in \alpha$ isotopic to some $b' \in \beta'$ if $e(b') = 0$ for all $e \in E_T(\alpha)$. If that is the case, we trace $S(T, b')$. Since $b' \cap a = \emptyset$, the curve a is still fully in the edges of the street complex. Nonetheless, a is isotopic to b' if and only if they cobound a cylinder [25], a case that can be determined in time $O(|S(T, b')|)$. Checking this for every pair of curves a and b' that do not intersect – there can be at most g^2 of these pairs – gives a test for the first hypothesis.

If the above test returns no clear reduction pair, we look for a separating curve, s, disjoint from α and β, which exists if and only if s is non-contractable in $\Sigma'' = \Sigma_g \backslash (\alpha \cup \beta)$. Because $\Sigma' = \Sigma_g \backslash \beta$ is a $2g$ puncture sphere, Σ'' is a union of (potentially one) punctured spheres, and

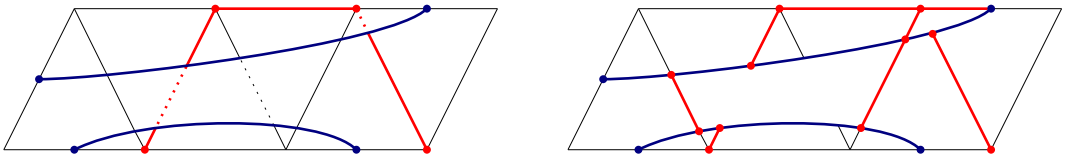


Figure 4 Diagram illustrating the proof of Theorem 6. Left: part of the overlay graph defined by the β curves (blue) with respect to a triangulation (black) and the corresponding embeddings of a segment of the α curves (red). Redundant ports are shown in dashed. A cellular embedding of the surface Σ'' can be obtained by puncturing the diagram along both blue and red curves. Right: another cellular embedding for Σ'' in which we enlarge the holes punctured when cutting α by moving redundant ports in $E_T(\alpha)$ to both ends of the streets.

it embeds a non-contractable curve if and only if one of its components is not a disk. This means that we only need to compute a cellular embedding of Σ'' and check its topological type. We start computing a triangulation for Σ' as in the proof of Theorem 5 by tracing $S(T,\beta)$ and disconnecting from the gluing rules the edges of $E_{S(T,\beta)}(\beta)$. We then want to further cut Σ' along α. But, because α is edged with respect to T, each segment of $E_T(\alpha)$ will be a port in the overlay graph of β (with respect to T). Each such a port will either be non-redundant – in which case it will still be an edge of $S(T,\beta)$) – or it will be redundant – in which case it will be in the crossing sequence of some non-degenerate street and will be deleted to form $S(T,\beta)$ (left side of Figure 4). Therefore, for each street in $E_{S(T,\beta)}(\beta)$ (degenerate or not), we can use Part (c) of Lemma 1 and Part (b) of Theorem 3 to determine, in time $O(\|E_T(\alpha)\| \times m|T|)$, if there is an occurrence of an edge in $E_T(\alpha)$ in a street's crossing sequence and delete from the complex $S(T,\beta)$ (the interior of) any such a street. Because the streets are disks in Σ', bounded by one or two segments of β, we note that, topologically, deleting the full interior of the streets that are transverse to some edge in $E_T(\alpha)$ is equivalent to puncturing a hole corresponding to a redundant port of the overlay graph by simply "enlarging" the hole (right side of Figure 4). This means that we have a cellular embedding of Σ'' which can be queried, in polynomial-time, for its topological type. Assuming that Σ'' has a component that is not a disk: we can use the techniques of [23] (refer to Section 2.3 of [14] for details) to find, in time $O(|T'|)$ a non-contractable curve s in Σ'' expressed as an SLP of complexity $O(m|T| + |T|) = O(m|T|)$. We finally compute $S(T,s)$, in time $O(m|T|^2)$, to check whether a diagram is defined on each component of $\Sigma_g \backslash s$. For such, we need only to find 1. the number of α and β components on each side of s; and, 2. whether s is not isotopic to an α or β curve by again looking for bounded cylinders. ◀

▶ **Theorem 7** (`GET` $H_1(M)$)**.** *The Smith normal form of the presentation matrix of $H_1(M)$ can be found in time $\widetilde{O}(m^2g^3 + m|T|)$, where $\widetilde{O}$ indicates some hidden polylogarithmic factors.*

Proof. We first compute the presentation matrix K of the homology group whose coefficients are algebraic intersection numbers between the diagram's curves. Because we assume that each a_j is a list of edges $E_T(a_j)$, given an SLP $I_T(b_i)$,

$$\hat{i}(b_i, a_j) = \sum_{e \in E_T(a_j)} e(I_T(b_i)) - e^{-1}(I_T(b_i)),$$

which, by Lemma 1, can be computed in time $O(\|I_T(b_i)\| \times \|E_T(a_j)\|)$. Therefore, referring to Theorem 3 to compute $I_T(\beta)$, the total computational time of this procedure is $O\Big(m|T| + \sum_{i=1}^{g}\sum_{j=1}^{g} \|I_T(b_i)\|\|E_T(a_j)\|\Big) = O(m|T| + m|T|)$. Finally, we reduce the matrix K to its Smith normal form in time $\widetilde{O}(g^3m^2)$ by the Kannan-Bachem algorithm [52]. ◀

▶ **Theorem 8** (`GET` $\pi_1(M)$)**.** *A presentation of* $\pi_1(M)$ *can be computed in time* $O(m|T|)$*.*

Proof. Given $I_T(\beta) = \{I_T(b)\}_{b\in\beta}$ delete, in linear time on the complexity of the SLPs, all references to edges $E - E_T(\alpha)$ and substitute each edge of $E_T(\alpha)$ by the symbol that represents its associated component $a \in \alpha$, here seen as a generator $\pi_1(M)$. This gives g distinct SLPs $r_i = [x_1^i = \text{EXPR}_1^i, \dots, x_{\ell_i}^i = \text{EXPR}_{\ell_i}^i]$ whose complexities add up to m and describe the relations of the presentation of $\pi_1(M)$. A presentation of $\pi_1(M)$ can then be defined with generators $a_1, \dots, a_g, x_1^1, \dots, x_{\ell_i-1}^1, \dots, x_{\ell_g-1}^g$ and relators $(x_j^i)^{-1} \cdot \text{EXPR}_j^i$ (this corresponds to repeated applications of a Tietze move [36] to the presentation of $\pi_1(M)$ computed using the method described at the end of Section 2.1). The output is a balanced presentation of $\pi_1(M)$ with $\|I_T(\beta)\|$-many generators and relators. The total time is dictated by Theorem 3. ◀

▶ Remark 9. Because the outputted presentation of $\pi_1(M)$ in Theorem 8 has linear size on the input, most complexity results about *uncompressed presentations* of 3-manifolds (such as triangulations) that appeal to the fundamental groups still hold in our *compressed* data structure. This is the case, for example, of Zentner's proof that sphere recognition lies in co-`NP` [55] (assuming the generalized Riemann hypothesis). Similarly, the usual bounds on the volume [16] and diameter [54] of hyperbolic 3-manifolds can be naturally translated to our notion of complexity.

5 Examples and experiments

Given a Heegaard word presentation (Σ_g, ϕ) of a closed 3-manifold M, there exist algorithms to compute, in linear time on the length of ϕ, a triangulation of M with as many tetrahedra [1, 2, 5, 38]. This implies, for example, that the first homology group of M, $H_1(M)$, can be computed in polynomial-time on $|\phi|$. These algorithms should be contrasted with Theorem 4, which says that SLPs of the intersection sequence $I_T(\beta = \phi(\alpha))$ can be computed in linear time on the complexity of ϕ *measured in power-notation form*. Because the fundamental group of M depends only on $I_T(\beta)$, algebraic information on splittings with significantly complicated gluing maps can be retrieved by Theorems 7 and 8.

Here, we explore the gains from the compressed data structure by comparing an early implementation of our methods with `SnapPy`'s [18] `Twister` [4] module, commonly used to compute triangulations from Heegaard words. Since experiments are limited to Heegaard words as input, there is no need to use proper normal coordinates or even triangulations of Σ_g; instead, we may compute the SLPs of the induced diagram with respect to general cellular embeddings, therefore avoiding the high cost of cyclic reductions. This guarantees that, for the problems analyzed, our theoretical complexity is at least equal to `Twister`'s. Even without these simplifying assumptions, we are still exponentially faster in the worst cases, as we show in the following section about lens spaces.

5.1 Genus 1

Heegaard splittings over the genus 1 surface – i.e., the torus – form the well-known family of *lens spaces*. Every lens space is uniquely given by a pair of integers $L(p, q)$, which, for a Heegaard splitting (Σ_g, a, b), represents the algebraic intersection numbers of b with the usual longitude and meridian of the torus (there is some ambiguity in the values of p, q, see [47]). A Dehn twist about the meridian m of the torus maps (p, q) to $(p + q, q)$, while about the longitude ℓ maps to $(p, q - p)$. This gives a representation of $\text{Mod}(\Sigma_1)$ in $\mathbb{Z}^2$ from which the homology of the Heegaard splitting (Σ_g, ϕ) can be exactly computed by $|\phi|$ matrix

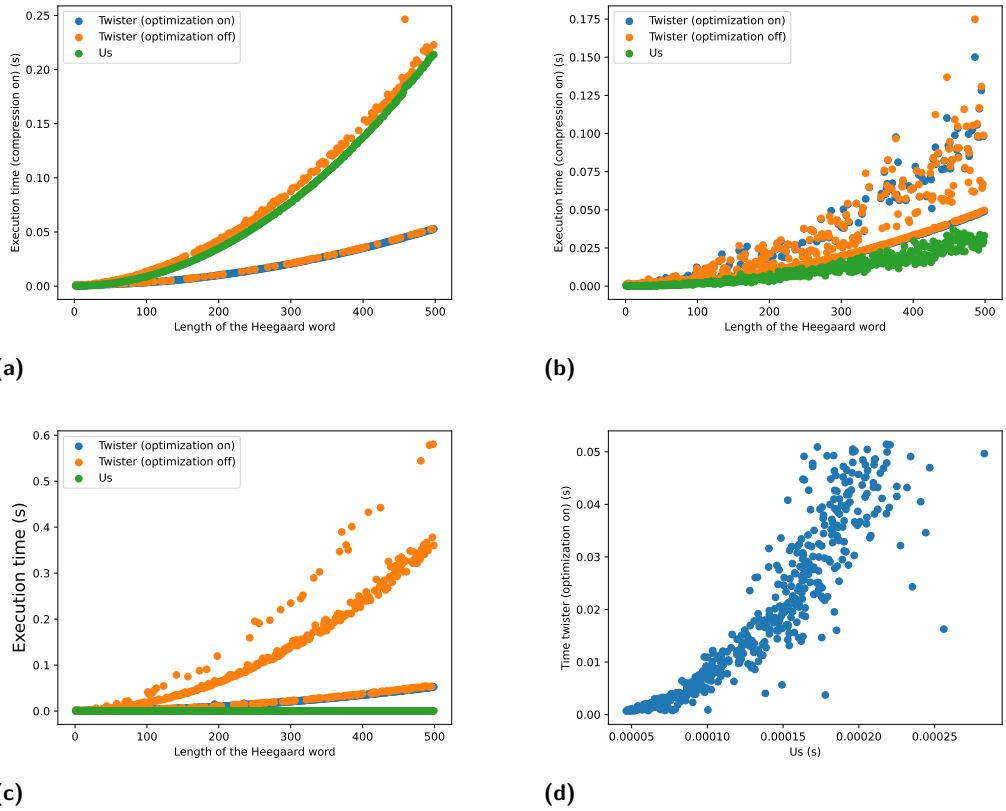


Figure 5 Experiments with computing the homology group H_1 of Heegaard splittings over the torus. Averages were computed for 30 repetitions.

multiplications and one query on the final value of q. We used this approach to benchmark the accuracy of our method, which returned the expected H_1 for input words ϕ of length up to 1000, the maximum size we tested.

Figure 5 compares the average computational time needed to find $H_1(\Sigma_1, \phi)$ taken by our method and by `Twister` for four different families of ϕ indexed by an integer n: (a) words of form $(\tau_m \tau_\ell^{-1})^n$; (b) uniformly sampled words of length n; (c) words of form τ_ℓ^n; and (d) words of form τ_ℓ^n, but this time, given in power-notation form. Averages were taken over 30 repetitions. In all cases, we compared our times with `Twister`'s with and without optimization (where, by optimization, the authors mean the greedy folding of tetrahedra; refer to their documentation for details[1]). Although optimization drastically reduces computational time, it seems to increase numerical error. In every instance, except for (d), we first transformed the words into a power-notation form before using them to compute $I_T(b)$; in total, this can be done in linear time on $|\phi|$.

▶ Remark 10. A direct application of the representation of $\mathrm{Mod}(\Sigma_1)$ implies that, for the family of case (a), $H_1^n = \mathbb{Z}/f_{2n}$ where f_n is the n-th Fibonacci number (starting at 0). We used this relation to benchmark the accuracy of the methods for different values of n. While our technique always agrees with the expected value for up to $n = 500$, `Twister` (with optimization both on and off) started deviating for values of $n \geq 25$. Interestingly, we did not observe divergences for higher genii splittings.

[1] Available at `https://snappy.computop.org`.

Case (a) is where our advantages compared to `Twister` are the worst. This is because the words $(\tau_m \tau_\ell^{-1})^n$ do not admit a non-trivial power-notation form, which is mainly responsible for our gains in the other cases. Nevertheless, it is worth highlighting that, while `Twister` is partially implemented in `C++`, our code is fully `python`. Case (b) represents the average case of computing the homology of lens spaces given a gluing map ϕ. We notice that we are now consistently faster for all values of n tested. This happens exactly as we explore redundancies by first putting the word in power-notation form, with significant gains every time the same Dehn twist occurs in sequence (since $\mathrm{Mod}(\Sigma_g)$ has two generators, this will be the case quite often). The gains due to repetitions are further highlighted in (c): while `Twister` is bound to take $O(n^c)$, for some $c > 1$, to compute the homologies, our approach takes time $O(n + \lg n)$, where the linear dependence comes solely from compressing the inputs to power-notation. In fact, if we assume the words already in power-notation, we are exponentially faster than `Twister`, as shown in case (d).

We can actually explore our structure's usage of power-notation to compute invariants of families of 3-manifolds that would necessarily require exponentially many tetrahedra to be represented. The minimum number of tetrahedra in a *layered triangulation* of a lens space $L(p, q)$ is $\sum_i (a_i) - 1$, where $(a_0, \ldots, a_k)$ is the continued fraction expansion of q/p [34]. Moreover, [32, 35] discovered families of lens spaces – including the spaces defined by $p = 2n, q = 1$ and $p = 4n, q = 2n - 1, n \geq 2$ – for which layered triangulations are minimal among *all* triangulations. So, for example, while the family $L(2^n, 1)$ necessarily requires exponentially many tetrahedra to be represented, it can be encoded in linear space in our data structure. It seems natural to expect that there are many other families of manifolds, with diagrams exponentially smaller than the minimum number of tetrahedra, for which our techniques can provide real computational gains compared to any existing method.

References

1. Gorjan Alagic and Edgar A. Bering. Quantum algorithms for invariants of triangulated manifolds. *Quant. Inf. Comput.*, 12(9-10):0843–0863, 2012. doi:10.26421/QIC12.9-10-8.
2. Gorjan Alagic and Catharine Lo. Quantum invariants of 3-manifolds and NP vs #P. *Quantum Info. Comput.*, 17(1–2):125–146, 2017.
3. M. Aschenbrenner, S. Friedl, and H. Wilton. *3-manifold Groups*. EMS series of lectures in mathematics. European Mathematical Society, 2015. URL: https://books.google.ch/books?id=syEEkui2AkoC.
4. Mark Bell, Tracy Hall, and Saul Schleimer. Twister (computer software). https://bitbucket.org/Mark_Bell/twister/, 2008–2014. Version 2.4.1.
5. Peter Brinkmann and Saul Schleimer. Computing triangulations of mapping tori of surface homeomorphisms. *Experimental Mathematics*, 10(4):571–581, 2001. doi:10.1080/10586458.2001.10504677.
6. Rhuaidi Antonio Burke, Benjamin A. Burton, and Jonathan Spreer. Small Triangulations of 4-Manifolds and the 4-Manifold Census. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:16, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2025.28.
7. Benjamin A. Burton. Efficient enumeration of 3-manifold triangulations. *Australian Mathematical Society Gazette*, 31(2):108–114, 2004.
8. Benjamin A. Burton. A new approach to crushing 3-manifold triangulations. In *Proceedings of the twenty-ninth annual symposium on Computational geometry*, pages 415–424, 2013. doi:10.1145/2462356.2462409.
9. Benjamin A. Burton. The cusped hyperbolic census is complete. *arXiv preprint*, 2014. arXiv:1405.2695.

10 Benjamin A. Burton. Tabulation of knots & 3-manifolds. Research Data Australia, University of Queensland, 2020. Dataset, https://researchdata.edu.au/tabulation-knots-3-manifolds/3370083.

11 Benjamin A. Burton. The Next 350 Million Knots. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2020.25.

12 Benjamin A. Burton, Ryan Budney, William Pettersson, et al. Regina: Software for low-dimensional topology. http://regina-normal.github.io/, 1999–2023.

13 Benjamin A. Burton and William Pettersson. An edge-based framework for enumerating 3-manifold triangulations. In *31st International Symposium on Computational Geometry (SoCG 2015)*, pages 270–284. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.SOCG.2015.270.

14 Erin W Chambers, Éric Colin De Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey. Splitting (complicated) surfaces is hard. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 421–429, 2006. doi:10.1145/1137856.1137918.

15 John H Conway. An enumeration of knots and links, and some of their algebraic properties. In *Computational problems in abstract algebra*, pages 329–358. Elsevier, 1970.

16 Daryl Cooper. The volume of a closed hyperbolic 3-manifold is bounded by times the length of any presentation of its fundamental group. *Proceedings of the American Mathematical Society*, 127(3):941, 1999.

17 Francesco Costantino, Yang-Hui He, Elli Heyes, and Edward Hirst. Learning 3-manifold triangulations. *Journal of Physics A: Mathematical and Theoretical*, 2024.

18 Marc Culler, Nathan M. Dunfield, Matthias Goerner, and Jeffrey R. Weeks. SnapPy, a computer program for studying the geometry and topology of 3-manifolds. Available at http://snappy.computop.org (19/05/2025).

19 Basudeb Datta. Minimal triangulations of manifolds. *arXiv preprint math/0701735*, 2007.

20 Olivier Devillers, Jean-Daniel Boissonnat, Mariette Yvinec, and Monique Teillaud. Triangulations in CGAL. In *Proceedings of the 16th Annual Symposium on Computational Geometry*, pages 11–18. ACM, 2000.

21 Clifford H Dowker and Morwen B Thistlethwaite. Classification of knot projections. *Topology and its Applications*, 16(1):19–31, 1983.

22 Henrique Ennes and Clément Maria. Hardness of computation of quantum invariants on 3-manifolds with restricted topology, 2025. doi:10.48550/arXiv.2503.02814.

23 Jeff Erickson and Sariel Har-Peled. Optimally cutting a surface into a disk. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 244–253, 2002. doi:10.1145/513400.513430.

24 Jeff Erickson and Amir Nayyeri. Tracing compressed curves in triangulated surfaces. *Discrete Computational Geometry*, 49(4):823–863, June 2013. doi:10.1007/s00454-013-9515-z.

25 Benson Farb and Dan Margalit. *A primer on mapping class groups*, volume 41. Princeton University Press, 2011.

26 Torben Hagerup. Sorting and searching on the word RAM. In *STACS 98: 15th Annual Symposium on Theoretical Aspects of Computer Science Paris, France, February 25–27, 1998 Proceedings 15*, pages 366–398. Springer, 1998. doi:10.1007/BFB0028575.

27 Wolfgang Haken. Über das homöomorphieproblem der 3-mannigfaltigkeiten. i. *Mathematische Zeitschrift*, 80(1):89–120, 1962.

28 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

29 Alexander He, James Morgan, and Em K Thompson. An algorithm to construct one-vertex triangulations of Heegaard splittings. *arXiv preprint*, 2023. arXiv:2312.17556.

30 John Hempel. *3-Manifolds*, volume 349. American Mathematical Society, 2022.

31 William Jaco and Ulrich Oertel. An algorithm to decide if a 3-manifold is a Haken manifold. *Topology*, 23(2):195–209, 1984.

32 William Jaco, Hyam Rubinstein, and Stephan Tillmann. Minimal triangulations for an infinite family of lens spaces. *Journal of Topology*, 2(1):157–180, 2009.

33 William Jaco and J Hyam Rubinstein. 0-efficient triangulations of 3-manifolds. *Journal of Differential Geometry*, 65(1):61–168, 2003.

34 William Jaco and J Hyam Rubinstein. Layered-triangulations of 3-manifolds. *arXiv preprint math/0603601*, 2006.

35 William Jaco, J Hyam Rubinstein, and Stephan Tillmann. Coverings and minimal triangulations of 3–manifolds. *Algebraic & Geometric Topology*, 11(3):1257–1265, 2011.

36 David Lawrence Johnson. *Presentations of groups*. Cambridge university press, 1997.

37 Jesse Johnson. Notes on Heegaard splittings. *preprint*, 2006.

38 Robert Koenig, Greg Kuperberg, and Ben W Reichardt. Quantum computation with Turaev–Viro codes. *Annals of Physics*, 325(12):2707–2749, 2010.

39 Greg Kuperberg. Identifying lens spaces in polynomial time. *Algebraic & Geometric Topology*, 18(2):767–778, 2018.

40 Marc Lackenby. Algorithms in 3-manifold theory, 2020. `arXiv:2002.02179`.

41 Marc Lackenby. Some fast algorithms for curves in surfaces. *arXiv preprint*, 2024. `doi:10.48550/arXiv.2401.16056`.

42 Marc Lackenby and Jessica S Purcell. The triangulation complexity of elliptic and Sol 3-manifolds. *Mathematische Annalen*, 390(2):1623–1667, 2024.

43 William BR Lickorish. A finite set of generators for the homeotopy group of a 2-manifold. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 60, pages 769–778. Cambridge University Press, 1964.

44 Sergei Vladimirovich Matveev and SV Matveev. *Algorithmic topology and classification of 3-manifolds*, volume 9. Springer, 2007.

45 Masamichi Miyazaki, Ayumi Shinohara, and Masayuki Takeda. An improved pattern matching algorithm for strings in terms of straight-line programs. In *Combinatorial Pattern Matching: 8th Annual Symposium, CPM 97 Aarhus, Denmark, June 30–July 2, 1997 Proceedings 8*, pages 1–11. Springer, 1997. `doi:10.1007/3-540-63220-4_45`.

46 Edwin E Moise. Affine structures in 3-manifolds: V. the triangulation theorem and hauptvermutung. *Annals of mathematics*, 56(1):96–114, 1952.

47 Nikolai Saveliev. *Lectures on the topology of 3-manifolds: an introduction to the Casson invariant*. Walter de Gruyter, 2011.

48 Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Algorithms for normal curves and surfaces. In *Computing and Combinatorics: 8th Annual International Conference, COCOON 2002 Singapore, August 15–17, 2002 Proceedings 8*, pages 370–380. Springer, 2002. `doi:10.1007/3-540-45655-4_40`.

49 Marcus Schaefer, Eric Sedgwick, and Daniel Stefankovic. Computing Dehn twists and geometric intersection numbers in polynomial time. In *CCCG*, volume 20, pages 111–114, 2008.

50 Saul Schleimer. Sphere recognition lies in np. *Low-dimensional and symplectic topology*, 82:183–213, 2011.

51 Jennifer Schultens. *Introduction to 3-manifolds*, volume 151. American Mathematical Soc., 2014.

52 Francis Sergeraert. About the Kannan-Bachem algorithm, 2024. `doi:10.48550/arXiv.2411.02422`.

53 Daniel Štefankovič. *Algorithms for simple curves on surfaces, string graphs, and crossing numbers*. Phd thesis, University of Chicago, 2005. Available at `http://people.cs.uchicago.edu/~laci/students/stefankovic-phd.pdf`.

54 Matthew E White. A diameter bound for closed, hyperbolic 3-manifolds. *arXiv preprint*, 2001. `arXiv:math/0104192`.

55 Raphael Zentner. Integer homology 3-spheres admit irreducible representations in sl(2,c). *Duke Mathematical Journal*, 167(9), June 2018. `doi:10.1215/00127094-2018-0004`.

Unlabeled Multi-Robot Motion Planning with Improved Separation Trade-Offs

Tsuri Farhana
The Stein Faculty of Computer and Information Science, Ben-Gurion University of the Negev, Beer Sheva, Israel

Omrit Filtser
Department of Mathematics and Computer Science, The Open University of Israel, Ra'anana, Israel

Shalev Goldshtein
Department of Mathematics and Computer Science, The Open University of Israel, Ra'anana, Israel

Abstract

We study unlabeled multi-robot motion planning for unit-disk robots in a polygonal environment. Although the problem is hard in general, polynomial-time solutions exist under appropriate separation assumptions on start and target positions. Solovey et al. (RSS'15) provide a near-optimal solution assuming that start/target positions must have pairwise distance at least 4, and at least $\sqrt{5} \approx 2.236$ from obstacles. This raises the question of whether polynomial-time algorithms can be obtained in even more densely packed environments.

In this paper we present a generalized algorithm that achieve different trade-offs on the robots-separation and obstacles-separation bounds, all significantly improving upon the state of the art. Specifically, we obtain polynomial-time constant-approximation algorithms to minimize the total path length when (i) the robots-separation is $2\frac{2}{3}$ and the obstacles-separation is $1\frac{2}{3}$, or (ii) the robots-separation is ≈ 3.291 and the obstacles-separation ≈ 1.354. Additionally, we introduce a different strategy yielding a polynomial-time solution when the robots-separation is only 2, and the obstacles-separation is 3. Finally, we show that without any robots-separation assumption, obstacles-separation of at least 1.5 may be necessary for a solution to exist.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases multi-robot motion planning

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.43

Related Version *Full Version*: `https://arxiv.org/abs/2603.19502` [10]

Funding *Omrit Filtser*: This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 2135/24).
Shalev Goldshtein: This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 2135/24).

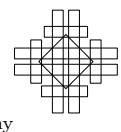
Acknowledgements We thank Dan Halperin for introducing us to the problem and for helpful discussions. We are also grateful to the anonymous reviewers for their insightful comments, which helped improve the structure and presentation of the paper.

1 Introduction

In the classic multi-robot motion-planning (MRMP) problem, a set of robots operate in a common workspace, and need to reach a set of target positions. More precisely, given a set of m starting positions and a set of m target positions, the goal is to compute a motion strategy for the robots to reach the targets while avoiding collisions with the boundary of the workspace and with other robots. In this paper we consider unit-disk robots that are moving in a polygonal workspace in the plane. If each robot is assigned to a specific target, the problem is referred to as *labeled* MRMP, and otherwise it is *unlabeled* MRMP.


© Tsuri Farhana, Omrit Filtser, and Shalev Goldshtein;
licensed under Creative Commons License CC-BY 4.0





The MRMP problem and its variants have been widely investigated, both from theoretical and practical perspectives (see, e.g., [3, 4, 5, 6, 9, 11] for some of the more recent works on the problem). Very recently, both the labeled and unlabeled variants of MRMP for unit disks in polygons with holes were shown to be PSPACE-hard [1]. Many other variants of the problem are known to be hard as well, see e.g. [8, 12, 14, 16].

Interestingly, the separation between the robots plays a key role in the difficulty of the problem. Adler, de Berg, Halperin, and Solovey [2] show that in a simple polygonal workspace with n vertices and m unit-disk robots, when the ***robots-separation***, denoted ρ, is 4 (i.e., all the start and target positions are at least distance 4 apart), a solution for unlabeled MRMP always exists, and can be found in $\tilde{O}(mn + m^2)$ time. In [7] this result is further improved by distinguishing between ***monochromatic*** robots-separation (i.e. the distance between two start or two target positions), and ***bichromatic*** robots-separation (i.e. the distance between a start position and a target position). The algorithmic approach in both papers [7, 2] is similar and does not provide any reasonable approximation on the length of the solution, i.e., the total length of the paths in the motion plan.

Solovey, Yu, Zamir, and Halperin [15] present a different approach for unlabeled MRMP, with optimality guarantee even in polygons with holes, but at the cost of adding an additional assumption on the distance between the obstacles to any start and target position. Specifically, they consider the case when the robots-separation is $\rho = 4$, and the ***obstacles-separation***, denoted ω, is $\sqrt{5} \approx 2.236$. Under these assumptions, they present an algorithm that runs in $\tilde{O}(m^4 + n^2m^2)$ time and returns a solution of total length at most $\mathrm{OPT} + 4m$. Here and throughout the paper, OPT denotes the minimum possible sum of lengths of the paths in a collision-free motion plan.

Notice that when assuming a large robots-separation ($\rho \geq 4$), the overall "density" of the robots inside the workspace becomes small, which may require the workspace to be much larger than the number of robots operating in it. Also, assuming a large obstacles-separation requires adding an additional buffering "layer" around the obstacles. This motivates the question of whether we can find weaker assumptions on either the robots-separation or the obstacles-separation, that are sufficient for having a polynomial time algorithm for unlabeled MRMP. Surprisingly, in this paper we show that both the robots-separation and obstacles-separation assumptions can be significantly relaxed, while still enabling a polynomial-time algorithm that achieves a constant-factor approximation of the solution length.

In previous works, namely [2, 7, 15], the algorithmic approach uses a ***monotone*** strategy, where robots move to their target position one by one according to some order, while other robots must remain at their start/target positions. The main result in our paper uses a weakly-monotone strategy. In a ***weakly-monotone*** motion plan, each start/target position p is associated with a revolving area - a disk of radius $r > 1$ that contains the unit disk centered at p and is empty of obstacles and other unit disks centered at other start/target positions. The robots move to their target position one by one according to some order, while other robots must remain inside the revolving area associated with their current position. Such a weakly-monotone strategy was used, e.g., in [4, 13] for a variant of labeled MRMP with unit-disk robots.

Our contribution. In this paper we present two different algorithms for unlabeled MRMP, that provide tighter trade-offs between robot-separation and obstacles-separation assumptions. In particular, for a parameter $0 \leq \varepsilon \leq 1$, a polygonal environment of complexity n, and m unit-disc robots, our first algorithm computes in $\tilde{O}(m^4 + n^2m^2)$ time a collision-free motion plan of total length $O(\mathrm{OPT})$, assuming that $\rho = \max\{4 - 2\varepsilon, 2 + \varepsilon\}$ and $\omega =$

$\max\{1+\varepsilon, \sqrt{(3-\sqrt{3}-\varepsilon)^2+1}\}$ (see Theorem 9). In Table 1 we highlight three interesting values of ε that minimizes either ρ, ω or the guaranteed length of the solution. This algorithm can be viewed as a generalization of the algorithm in [15], and indeed, as a consequence of our analysis we positively resolve a conjecture made in [15], by showing that $\omega = 1.614$ is sufficient when $\rho = 4$. Moreover, in the full version of our paper [10] we show that a monotone motion plan (such as the one produced by the algorithm in [15]) may not always exist when $\omega < 1.614$, and therefore to achieve smaller separation bounds one must use a different strategy. Indeed, as mentioned above, our motion plan is weakly-monotone. Interestingly, we also show that a weakly-monotone motion plan may not always exist when $\omega < 1.354$.

Note that our first algorithm does not cover the case of $\rho = 2$, which is the optimal (monochromatic) robots-separation. We therefore present a second algorithm using a completely different approach, which given a simple polygon with n vertices and m unit-disk robots, computes in $O(m^3 + mn^2)$ time a collision-free motion plan of total length $\mathrm{OPT} + O(m^2)$, assuming that $\rho = 2$ and $\omega = 3$ (see Theorem 10). Finally, we show that $\omega = 1.5$ may be necessary for a solution to exist.

Table 1 Summary of algorithms and their separation assumptions.

ω	ρ	Solution length	Running time	Notes	Reference
1	4	no guarantee	$\tilde{O}(mn + m^2)$	Simple polygons	[2, 7]
$\sqrt{5}$	4	$OPT + 4m$	$\tilde{O}(m^4 + n^2m^2)$		[15]
≈ 1.614	4	$OPT + 4m$	$\tilde{O}(m^4 + n^2m^2)$		Theorem 9
≈ 1.354	≈ 3.291	$O(OPT)$	$\tilde{O}(m^4 + n^2m^2)$		Theorem 9
$1\frac{2}{3}$	$2\frac{2}{3}$	$O(OPT)$	$\tilde{O}(m^4 + n^2m^2)$		Theorem 9
3	2	$OPT + O(m^2)$	$\tilde{O}(m^3 + mn^2)$	Simple polygons	Theorem 10

Technical ideas and overview of the paper

Our approach in the first algorithm generalizes the strategy of Solovey et al. [15] for unlabeled MRMP, which is briefly described as follows. Given a set S of start positions, a set T of target positions, and a polygonal workspace $\mathcal{W}$, first compute an optimal set Γ of paths assigning each starting point to a target point (a perfect matching). This set of paths minimizes the sum of geodesic distances, when ignoring the other robots in the workspace. Then, one can show that when assuming $\rho = 4$ and $\omega = \sqrt{5}$, there is always at least one target which is a so-called standalone goal, that is, a target t that does not block any other path in Γ. Such a target is useful because we can place a robot there and then treat it as an obstacle. However, there might be some other robot that blocks the path that leads to t, in which case we can make a switch: we push the last robot that blocks this path to t. This switching process is possible because of the assumption $\omega = \sqrt{5}$. Then, run the algorithm recursively, but after removing the start and target positions that where satisfied in the previous recursive step, and updating the workspace to treat the robot at t as an obstacle.

Our weakly-monotone motion plan is obtained by inserting a very important modification to the algorithm of [15]: instead of choosing a standalone target in each iteration, we relax the requirement and require only an "almost standalone" target – a target that may interrupt

other paths, but only by a small amount. In our strategy, we distinguish between "blocking" and "interrupting" positions. Instead of always switching paths when another robot is standing in the way, we allow robots that interrupt a path by only a small amount to move away from the path. A robot that interrupts a given path may move slightly in its close neighborhood (a disk of radius equal to the amount of interruption plus one) in order to clear the way. In contrast, a robot that blocks a path may not have enough wiggle room, and we will switch the paths.

As suggested above, this combined strategy can be viewed as a generalization of [15]: we introduce another parameter, called the "overlap parameter", which is used to determine whether a position is blocking a path or only interrupting it. This parameter allows us to define a set of constraints on ρ and ω, that allows (i) the existence of an interrupting target, (ii) the option to switch paths in the case when a robot is blocking a path, and (iii) the existence of a sufficiently large free neighborhood for an interrupting robot to clear the way. It turns out that there is an interesting trade-off between these constraints, leading to the main result of our paper (Theorem 9).

Note that the concept of interrupting positions raises some issues that were not present in the algorithm of [15]. For example, in each iteration of the recursive algorithm we need to take into account that the robot placed on a target may still interrupt other paths. In [15] this is solved simply by removing a unit disk around the interrupting target from the workspace. However, in our case, such a robot may still need to move in order to clear the way for other robots. Moreover, if we simply treat it as an obstacle, the assignment of paths in the next iteration may be too large. We solve this issue by observing that it is actually enough to remove a smaller disk around a settled target when computing the paths, while still allowing it to move within its small neighborhood when planning the motion in the following iterations. Another issue is that there may be two or more robots that interrupt a path and may need to move simultaneously. Here we describe a simple motion with respect to the ray connecting the robot moving on the path with the interrupting robots.

In Section 3, we systematically define and analyze the constraints that are required for each part of this strategy, each constraint depends on the overlap parameter. In Section 4 we present the complete generalized algorithm and prove its correctness. Then, we show how to choose an overlap parameter and adjust constraints to get the minimum possible obstacles-separation and the minimum possible robots-separation.

In our first algorithm, to clear a path, we only move robots that intersect that path. Such a strategy cannot work when the robots-separation is exactly 2, because a robot that intersect a path may be block by other robots that do not intersect that path, and thus will not be able to move and clear the path. Therefore, to achieve a separation bound of 2 between the robots, we present in Section 5 a completely different strategy, which we call the Exodus algorithm. In this strategy, we also iteratively pick a path between start and target positions that were not yet satisfied, and move a robot along that path, however, in contrast to the first algorithm, we clear the path completely by also moving robots that do not intersect the path. In simple polygons, this strategy allows us to assume robots-separation of only 2, with obstacles-separation of 3, and it applies also for labeled MRMP (see Theorem 10).

Note that the monochromatic separation must be at least 2, so the robots-separation is almost optimal (the bichromatic separation can be 0). In Section 6 we preset lower bounds on the obstacles-separation, and we leave open the question of what is the upper bound on ω when robots-separation is optimal.

2 Preliminaries

We consider a set of m unit-disk robots, moving in a polygonal ***workspace*** $\mathcal{W} \subseteq \mathbb{R}^2$ (which may be simple or cluttered with polygonal obstacles), whose overall number of edges is n. Below, we mostly follow the notations in [7] and [15].

The ***obstacle space*** $\mathcal{O}$ is the complement of the workspace $\mathcal{W}$, that is, $\mathcal{O} = \mathbb{R}^2 \setminus \mathcal{W}$. We refer to the points in $\mathcal{W}$ as positions, and say that a robot is at position $x \in \mathcal{W}$ when its center is positioned at the point x. For a point $x \in \mathbb{R}^2$ and a radius $r \in \mathbb{R}_+$, we define $D_r(x)$ to be the open disk of radius r centered at x. The unit-disk robots are defined to be open sets, and therefore two robots collide if and only if the distance between their positions is strictly less than 2. In addition, a robot collides with the obstacle space $\mathcal{O}$ if and only if its center is at distance strictly less than 1 from $\mathcal{O}$. The set of all positions in $\mathcal{W}$ where a unit-disk robot does not collide with the obstacle space is called the ***free space***, denoted $\mathcal{F} = \{x \in \mathbb{R}^2 \mid D_1(x) \cap \mathcal{O} = \emptyset\}$. Note that the free space is a closed set, and denote by $\partial\mathcal{F}$ the boundary of $\mathcal{F}$.

In the multi-robot motion-planning problem, we are given a workspace $\mathcal{W}$ with n edges and two sets $S = \{s_1, \ldots, s_m\}$ and $T = \{t_1, \ldots, t_m\}$ of points in the free space $\mathcal{F}$. The goal is to plan a collision-free motion for m robots, each positioned on a different starting point in S, such that by the end of the motion every target position in T is occupied by some robot. When the robots are distinguishable (i.e. labeled), the robot that starts at position s_i is required to end up at the target t_i. Otherwise, when the robots are indistinguishable (i.e., unlabeled), it does not matter which robot ends up at which target position, as long as all targets are occupied at the end of the motion.

Formally, we wish to find a set Γ of continuous paths $\gamma_1, \ldots, \gamma_m$, where $\gamma_i : [0,1] \to \mathcal{F}$ for every $1 \leq i \leq m$, and such that $\bigcup_{i=1}^m \gamma_i(0) = S$ and $\bigcup_{i=1}^m \gamma_i(1) = T$. In addition, the paths need to be collision-free, i.e., at any point in time $t \in [0,1]$, there are no i, j such that $\gamma_i(t)$ and $\gamma_j(t)$ are at distance strictly less than 2. Denote the size of a solution Γ by $|\Gamma| = \sum_{\gamma_i \in \Gamma} |\gamma_i|$, where $|\gamma|$ is the length of a path γ in the L_2 norm. In the optimization version of the problem, we are interested in finding a solution Γ which minimizes $|\Gamma|$, i.e., minimizing the sum of lengths of the paths.

Separation assumptions. We denote by ω the ***obstacles-separation*** assumption on an instance of MRMP, i.e., we assume that for every $p \in S \cup T$ and every point $x \in \mathcal{O}$, it holds that $\|x - p\| \geq \omega$. We denote by ρ the ***robots-separation*** assumption, i.e., we assume that for every $p_1, p_2 \in S \cup T$ it holds that $\|p_1 - p_2\| \geq \rho$.

Overlapping, blocking and interrupting positions

Let $p, q \in \mathcal{F}$ be two points in the free space. We define the ***overlap*** of p and q to be $\max\{0, 2 - \|p - q\|\}$ (see Figure 1). If $\mathcal{D}_1(p) \cap \mathcal{D}_1(q) = \emptyset$, then their overlap is 0. Otherwise, we say that p, q are ***ε-overlapping*** for $\varepsilon = 2 - \|p - q\|$, and note that $\|p - q\| = 2 - \varepsilon$.

We say that a position $p \in \mathcal{F}$ is ***ε-blocking*** a path γ if there exists $0 \leq t \leq 1$ such that the overlap of p and $\gamma(t)$ is **strictly larger than ε**. Blocking robots (i.e. robots that are placed on blocking positions) may be impossible to clear from the path, and we will have to make a "switch". We say that a point $p \in \mathcal{F}$ is ***ε-interrupting*** a path γ if for every $0 \leq t \leq 1$, the overlap of p and $\gamma(t)$ is **at most ε** (that is, it is not ε-blocking). Interrupting robots (i.e. robots that are placed on interrupting positions) will have enough wiggle room to move away from γ and "clear" the path. Notice that if for every $0 \leq t \leq 1$ the overlap of p

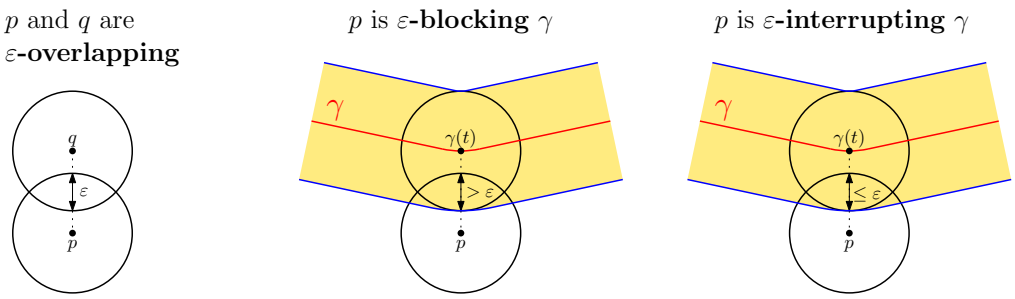


Figure 1 Overlapping, blocking, and interrupting positions.

and $\gamma(t)$ is 0, then for every $0 \leq t \leq 1$ it holds that $\|\gamma(t) - p\| > 2$, and therefore in this case we say that p is 0-interrupting γ. We say that a path γ is ***ε-blocked*** if there exists $s \in S$ that ε-blocks it, and that γ is ***ε-interrupted*** if no $s \in S$ is ε-blocking it.

The overlap parameter. For the analysis in Section 3, we fix a parameter $0 \leq \varepsilon \leq 1$ which we call the ***overlap parameter***. In these sections, for simplicity of the presentation, we sometimes omit ε from the above notions and just say "blocking" or "interrupting". In our analysis, we introduce a set of constrains described as functions of ε, which are required for the different parts of our algorithm to work. In Section 4 we show how to choose ε in order to optimize different separation bounds.

3 Constraints for tighter separation bounds

To give an intuition for the required constraints, we first provide a high-level overview of our strategy. The algorithm is recursive, and its input is the overlap parameter ε, a set S of starting positions, a set T of target positions, and a workspace $\mathcal{W}$. In each recursive iteration, one robot is moving to a target position, then S,T and $\mathcal{W}$ are updated accordingly.

The first step is to compute an ***optimal-assignment path set***, as it is defined in [15]:

▶ **Definition 1.** *Let Γ be a set of m geodesic paths $\gamma_1, \ldots, \gamma_m$, where $\gamma_i : [0,1] \to \mathcal{F}$ for every $1 \leq i \leq m$, and such that $\bigcup_{i=1}^{m} \gamma_i(0) = S$, $\bigcup_{i=1}^{m} \gamma_i(1) = T$. We say that Γ is an optimal-assignment path set for $S, T, \mathcal{W}$ if it minimizes $\sum_{i=1}^{m} |\gamma_i|$ over all such path sets.*

Note that an optimal assignment path set Γ is not necessarily a feasible solution, because it ignores possible collisions between robots. This also means that the sum of paths in Γ is at most the length of an optimal solution to our MRMP problem. Let $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$ be an optimal-assignment path set for $S, T, \mathcal{W}$.

In Section 3.1 we present a useful lemma showing that a blocking robot can enter the geodesic path that it blocks. This lemma will be useful throughout the analysis, and it requires one constraint on the obstacles-separation, and another on the robots-separation.

The next step is to find a target $t \in T$ which does not block any path in Γ (other than the path γ from $s \in S$ to t). Then we can move a robot from s to t along γ, and it will not block other paths in the following iterations. Note that t may still interrupt other paths. In Section 3.2 we show that such a target always exists, when assuming an additional constraint on the robots-separation.

If there is a position $s' \in S$, which blocks the path γ, we need to make a switch, i.e., we construct a new path in $\mathcal{F}$ from s' to t, which is not blocked by any other robot. In Section 3.3 we show how to construct such a switch path γ', under all the constraints defined earlier. If there is no such blocking position s', we simply set $\gamma' \leftarrow \gamma$.

We now have a path γ' from some $s'' \in S$ (which may be either s or s') to the target t, which in not blocked by any robots, but may be interrupted. In Section 3.4 we show how to move a robot from s'' to t along γ', while all interrupting robots (either on start or target positions) move slightly in order to clear the path. For this we introduce two additional constraints, one on the obstacles-separation, and one on the robots-separation.

Finally, we prepare the input for the next recursive iteration, by removing s'' from S, t from T, and $D_{1-\varepsilon}(t)$ from $\mathcal{W}$. Removing $D_{1-\varepsilon}(t)$ from $\mathcal{W}$ ensures that the robots placed on t does not block any path that will be computed in any of the next iterations.

In Section 4 we present the complete algorithm based on the building blocks presented in this section, prove its correctness and analyze its running time and approximation factor.

3.1 Geodesics and blocking positions in the free space

We begin by introducing the constraints under which a blocking robot positioned on some $p \in S \cup T$ can enter a geodesic path $\gamma : [0,1] \to \mathcal{F}$ between some start and target positions, via a straight line segment connecting it to the path.

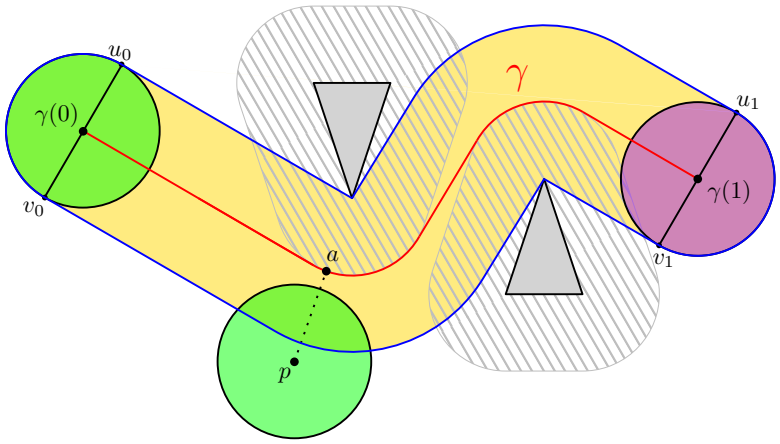


Figure 2 A geodesic path γ in $\mathcal{F}$ from a start position $\gamma(0) \in S$ to a target position $\gamma(1) \in T$ is illustrated in red. The obstacles are in gray, and the trace of γ is highlighted in yellow. A position p is blocking/interrupting γ. The point a is the closest point to p on γ.

A geodesic path between two positions $x, y \in \mathcal{F}$ is a shortest length path $\gamma : [0,1] \to \mathcal{F}$ where $\gamma(0) = x$ and $\gamma(1) = y$. Denote by $\mathcal{D}_1(\gamma) = \bigcup_{0 \le t \le 1} \mathcal{D}_1(\gamma(t))$ the ***trace*** of γ (see Figure 2). For $0 \le t_1 \le t_2 \le 1$, denote by $\gamma[t_1, t_2]$ the subpath of γ between $\gamma(t_1)$ and $\gamma(t_2)$.

In the full version of the paper [10], we provide some very useful geometric properties of geodesic paths in the free space, which result in the following constraints. Note that we have one constraint on the obstacles-separation, and another on the robots-separation.

▶ **Constraint 1.** $\omega \ge \sqrt{(3 - \sqrt{3} - \varepsilon)^2 + 1}$

▶ **Constraint 2.** [1] $\rho \ge \sqrt{\frac{1}{2}^2 + \left(3 - \frac{\sqrt{3}}{2} - \varepsilon\right)^2}$

[1] Note that later we introduce two more constraints on ρ that together make Constraint 2 redundant, however, Lemma 2 holds already for this weaker constraint and may be of independent interest.

Let $\gamma : [0,1] \to \mathcal{F}$ be a geodesic path. We say that a point a on γ is ***locally closest*** to p if there exists $\delta > 0$ such that $D_{\|p-a\|}(p) \cap \gamma[a-\delta, a+\delta] = a$.

▶ **Lemma 2.** *Let $\gamma : [0,1] \to \mathcal{F}$ be a geodesic path such that $\gamma(0), \gamma(1) \in S \cup T$, and let $p \in S \cup T$ be a position which is ε-blocking γ. Let $a = \gamma(t)$ be a point on γ which is locally closest to p and such that $\|p-a\| < 2-\varepsilon$. If Constraints 1 and 2 hold, then $\overline{pa} \subset \mathcal{F}$.*

Intuitively, Lemma 2 states that a robot blocking γ and positioned on p can move to a along the path $\overline{pa}$, as long as no other robots are standing in its way (see Figure 2).

3.2 Existence of an ε-interrupting target

We now define formally a generalization of the standalone goal defined in [15] (Definition 3). Our definition depends on the overlap parameter ε, and it is equivalent to a standalone goal for $\varepsilon = 0$. Let $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$ be an optimal-assignment path set for $S, T, \mathcal{W}$.

▶ **Definition 3.** *A target position $t \in T$ is an* ***ε-interrupting target****, if for every path $\gamma \in \Gamma$ such that $\gamma(1) \neq t$, it holds that t is not ε-blocking γ.*

To show that an ε-interrupting target always exists, we need an additional constraint.

▶ **Constraint 3.** $\rho \geq 4 - 2\varepsilon$

The following theorem is a generalization of Theorem 4 in [15]. The proof follows a similar logic, however, we need to take into account some issues that arise from the generalized definition and Constraints 1–3. See the full version [10] for a complete proof.

▶ **Theorem 4.** *Let Γ be an optimal-assignment path set for $S, T, \mathcal{W}$. If Constraints 1–3 hold, then there exists an ε-interrupting target.*

3.3 Dealing with a blocked geodesic path

Let $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$ be an optimal-assignment path set for $S, T, \mathcal{W}$, and assume w.l.o.g. that $\gamma_i(0) = s_i$ for every $1 \leq i \leq m$. Consider a geodesic path γ_i from s_i to t_j, and assume that is it ε-blocked. Our goal in this section is to describe the switching process: we find another start point $s_k \in S$ and a path $\gamma'_k \subset \mathcal{F}$ from s_k to t_j such that γ'_k is not ε-blocked. Moreover, we describe a path in $\mathcal{F}$ from s_i to t_ℓ, where $t_\ell = \gamma_k(1)$ (see Figure 3). The sum of lengths of these new paths will not be much longer than the original.

For a position $s_k \in S$ that ε-blocks γ_i, denote by $0 \leq w_{i,k} \leq 1$ the largest value such that $\gamma_i(w_{i,k})$ is the point on γ_i locally closest to s_k. We say that $s_k \in S$ is the ***last to ε-block*** γ_i if for any $s_x \in S \setminus \{s_k\}$ that ε-blocks γ_i, we have $w_{i,k} \geq w_{i,x}$ (i.e. the last locally closest point to s_k on γ_i is the closest to $\gamma_i(1)$ along γ_i).

Let s_k be the last to ε-block γ_i, and set $w = w_{i,k}$. Consider the paths $\gamma'_k = \overline{s_k\gamma_i(w)} \circ \gamma_i[w,1]$ and $\gamma'_i = \gamma_i[0,w] \circ \overline{\gamma_i(w)s_k} \circ \gamma_k$ (where γ_k is the geodesic path from s_k to t_ℓ, see Figure 3). We call γ'_i and γ'_k the ***switch paths*** of i and k, respectively. Notice that $\|s_k - \gamma_i(w)\| < 2-\varepsilon$, and therefore $|\gamma'_k| + |\gamma'_i| < |\gamma_k| + |\gamma_i| + 4 - 2\varepsilon$. The following theorem will help us in showing that the switch paths are not blocked (see [10] for a full proof).

▶ **Theorem 5.** *Assume that Constraints 1–3 hold. Then (i) both γ'_i and γ'_k are in $\mathcal{F}$, (ii) $\overline{s_k\gamma_i(w)}$ is not ε-blocked by any $p \in T \cup S \setminus \{s_k\}$, (iii) γ'_k is not ε-blocked by any $s_j \in S \setminus \{s_k\}$, and (iv) if t_j is not ε-blocking γ_k then it does not ε-block γ'_i.*

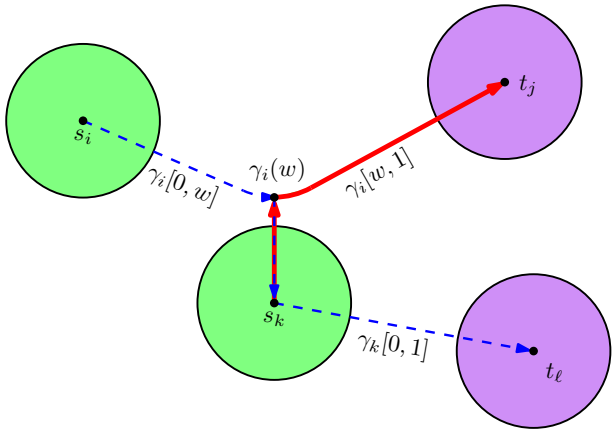


Figure 3 s_k is the last to ε-block γ_i. The switch paths are γ_i' (dashed blue) and γ_k' (red).

3.4 Dealing with robots interrupting a path

In this subsection, we discuss how to move a robot along an ε-interrupted path. We describe simple motion paths for the interrupting robots to clear the way, and we analyze the extra distance traveled by the interrupting robots. We note that shorter paths can be shown for the interrupting robots, however, as this does not effect the correctness of the algorithm, we chose to keep this part simple and describe a simple (yet rather short) motion path.

As mentioned at the beginning of this section, our motion plan consists of a weakly-monotone collision free path set, that is, robots moves to targets according some ordering one by one; when one robot travels from its starting position to some target, the other robots must stay in a small disk centered at their current position (which is either a start or a target position). To ensure that an interrupting robot is able to clear the way while remaining in a disk of radius $1+\varepsilon$ around a start/target position, we introduce two additional constraints.

▶ **Constraint 4.** $\omega \geq 1+\varepsilon$

▶ **Constraint 5.** $\rho \geq 2+\varepsilon$

Consider a robot A traveling in constant speed along a path γ_A from start position $s \in S$ to some target position $t \in T$. Let $p \in S \cup T$ be either a start or a target position, occupied by another robot B, which is ε-interrupting γ_A. We describe a motion plan for B by constructing a ***clearance path*** $\tilde{\gamma}_B : [0,1] \to \mathbb{R}^2$, along which the robot B will travel with constant speed, as follows.

Let $\gamma_A[w_1,w_2]$ be a maximal subpath where $\|\gamma_A(w) - p\| \leq 2$ for every $w \in [w_1,w_2]$, that is, when A travels along $\gamma[w_1,w_2]$, the robot B must clear the way. Note that there could be more than one such maximal subpath for B. We define the corresponding subpath $\tilde{\gamma}_B : [w_1,w_2] \to \mathbb{R}^2$ for B using a polar coordinate system, where p is the pole. Denote by r_x the radius and by θ_x the angle of a point x, and let $\tilde{\gamma}_B(w) := \big(2 - r_{\gamma_A(w)}, \theta_{\gamma_A(w)} + \pi\big)$.

The next lemma shows that the clearance path of B is feasible and relatively short. See [10] for the full proof.

▶ **Lemma 6.** *$\tilde{\gamma}_B(w)[w_1,w_2]$ is a continuous path that starts and ends at p, such that (i) $\tilde{\gamma}_B[w_1,w_2] \subset D_\varepsilon(p) \subset \mathcal{F}$, and (ii) $|\tilde{\gamma}_B[w_1,w_2]| \leq |\gamma_A[w_1,w_2]|$ for $0 \leq \varepsilon \leq 1$.*

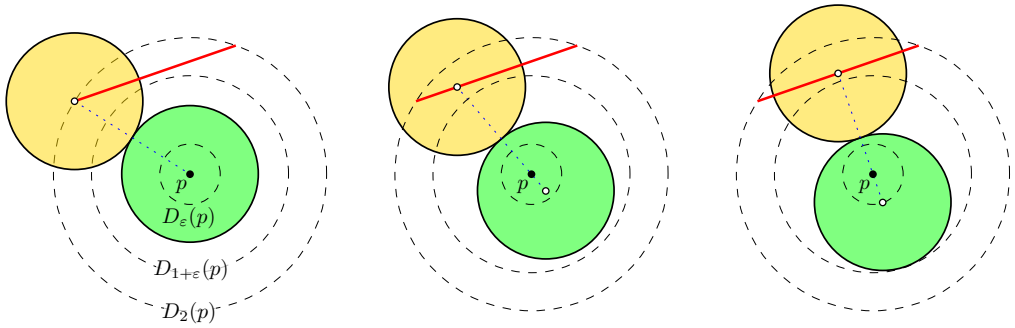


Figure 4 Left: the robot B is initially positioned at p. In red is a maximal subpath of γ_A where B needs to clear the way. Middle, right: the robot B move inside $\mathcal{D}_\varepsilon(p)$ while maintaining distance exactly 2 from the robot A.

Recall that B may interrupt γ_A in more than one maximal subpath, so overall we define the clearance path $\tilde{\gamma}_B : [0,1] \to \mathbb{R}^2$ as

$$\tilde{\gamma}_B(w) = \begin{cases} p, & \text{if } \|p - \gamma_A(w)\| \geq 2, \\ \left(2 - r_{\gamma_A(w)}, \theta_{\gamma_A(w)} + \pi\right), & \text{if } \|p - \gamma_A(w)\| < 2. \end{cases}$$

In addition, we construct such a clearance path for any other robot that interrupts γ_A. In the full version [10] we prove the following theorem, which shows that clearance paths are collision-free, assuming Constraints 1–5.

▶ **Theorem 7.** *Let A be a robot traveling in constant speed along a path γ_A from start position $s \in S$ to some target position $t \in T$, and let $\{\tilde{\gamma}_B \mid B \neq A\}$ be the set of clearance paths defined for the other robots. If Constraints 1–5 hold for $0 \leq \varepsilon \leq 1$, then the motion plan that corresponds to the paths $\gamma_A \cup \{\tilde{\gamma}_B \mid B \neq A\}$ is collision-free.*

4 Unlabeled MRMP with improved separation trade-offs

In this section we present and analyze the algorithm based on the tools developed in the previous section. The input for the algorithm is an overlap parameter ε, a workspace $\mathcal{W}$, a set S of starting positions and a set T of target positions, such that Constraints 1–5 hold. In addition, we assume that each connected component of $\mathcal{F}$ contains an equal number of start and target positions (as otherwise, there is no solution).

For the simplicity of the analysis, in this section we provide an iterative version of the algorithm. In the i'th iteration, for $1 \leq i \leq m$, the algorithm constructs a collision-free motion plan for all the robots over the time interval $[i-1, i]$, in which a single robot is moving from a start position to a target position along an ε-interrupted path (Section 3.3), and the other robots may move in their small neighborhood to clear the way (Section 3.4). The output path for a robot A will then be the concatenation $\gamma_A^* : [0, m] \to \mathcal{F}$ of the paths $\gamma_A : [i-1, i] \to \mathcal{F}$ constructed for it in all iterations. We then update S and T according to the path that was chosen, and remove from $\mathcal{W}$ a disk of radius $1 - \varepsilon$ around the target.

At the beginning of this section we gave a high level description of our algorithm, and a more detailed pseudo code is provided in Algorithm 1. Below, we prove that this algorithm returns a collision-free motion plan, with total length $O(\mathrm{OPT})$, where OPT is the optimal size of a solution to the problem.

Algorithm 1 Motion Plan for Unlabeled Disks.

Input: $\mathcal{W}$, S, T, and $0 \leq \varepsilon \leq 1$ such that Constraints 1–5 hold
Output: A collision-free motion plan

- For each robot $A \in [m]$, initialize an empty path γ_A^*.
- While $S \neq \emptyset$:
  1. Compute an optimal-assignment path set Γ for $S, T, \mathcal{W}$.
  2. Find an ε-interrupting target t_j, and let $\gamma_i \in \Gamma$ be the path from s_i to t_j.
  3. If γ_i is not ε-blocked, set $A \leftarrow i$, $s_A \leftarrow s_i$, $\gamma_A \leftarrow \gamma_i$.
  4. Else, let s_k be the last to ε-block γ_i, and set $A \leftarrow k$, $s_A \leftarrow s_k$, and $\gamma_A \leftarrow \gamma'_k$, where γ'_k is the switch path (Section 3.3).
  5. For all $B \neq A$, set $\gamma_B \leftarrow \tilde{\gamma}_B$, where $\tilde{\gamma}_B$ is the clearance path (Section 3.4).
  6. For each robot $B \in [m]$ set $\gamma_B^* \leftarrow \gamma_B^* \circ \gamma_B$.
  7. Set $\mathcal{W} \leftarrow \mathcal{W} \setminus \mathcal{D}_{1-\varepsilon}(t_j)$, $S \leftarrow S \setminus \{s_A\}$, $T \leftarrow T \setminus \{t_j\}$.
- Return the set of paths $(\gamma_1^*, \ldots, \gamma_m^*)$.

▶ **Theorem 8.** *Consider an input $\mathcal{W}$, S, T, and $0 \leq \varepsilon \leq 1$ such that Constraints 1–5 hold. If each connected component of $\mathcal{F}$ contains an equal number of start and target positions, then Algorithm 1 is guaranteed to return a collision-free motion plan Γ^*.*

Proof. First, observe that the solution returned by the algorithm is a path set that matches every start position in S to a unique target in T, because in each iteration we find a path between some $s \in S$ and $t \in T$ and then remove them from S and T for the next iteration.

Consider the first iteration of the algorithm. By Theorem 4, an ε-interrupting target t_j exists, and let γ_i be the path from s_i to t_j. If γ_i is ε-blocked, let $s_k \in S$ be the last to block γ_i, and by Theorem 5(iii), the switch path γ'_k is in $\mathcal{F}$, and it is not blocked by any other robot. Otherwise, γ_i is not blocked. In any case we choose an ε-interrupted path that leads to t_j, and by Theorem 7 the motion plan computed in this iteration is collision-free.

In the following iteration, after removing $D_{1-\varepsilon}(t_j)$ from $\mathcal{W}$, Constraints 1–5 still hold for the updated sets S and T: the robots-separation does not change, so Constraints 2, 3, and 5 still hold, and specifically by Constraint 3 for every $p_1, p_2 \in S \cup T$ we have $\|p_1 - p_2\| \geq 4 - 2\varepsilon$. Therefore, for every $p \in S \cup T \setminus \{t_j\}$ and $x \in D_{1-\varepsilon}(t_j)$ we have $\|p - x\| \geq 3 - \varepsilon$, which satisfies both Constraint 1 ($\|p - x\| \geq \rho_\varepsilon$) and Constraint 4 ($\|p - x\| \geq 1 + \varepsilon$), for any $0 \leq \varepsilon \leq 1$.

However, we also need to take into account the robot that is now positioned on t_j. The main observation that we will use is that t_j does not block any path γ in the free space of $\mathcal{W} \setminus D_{1-\varepsilon}(t_j)$; the reason is that the distance between any point on $a \in \gamma$ and a point $p \in D_{1-\varepsilon}(t_j)$ is at least 1, and therefore the $\|t_j - a\| \geq 2 - \varepsilon$ which means that the overlap between t_j and a is at most ε.

Consider an iteration $a > 1$ of the algorithm, and denote by $\mathcal{W}_a, S_a, T_a$ the sets $\mathcal{W}, S, T$ at the beginning of iteration a. The first step is to compute an optimal-assignment path set Γ_a. Let Γ_{a-1} be the optimal-assignment path set for $\mathcal{W}_{a-1}, S_{a-1}, T_{a-1}$. At the beginning of iteration a, we have $\mathcal{W}_a = \mathcal{W}_{a-1} \setminus \mathcal{D}_{1-\varepsilon}(t_j)$ and $T_a = T_{a-1} \setminus \{t_j\}$, where t_j is an ε-interrupting target in Γ_{a-1}. Since t_j does not block any path in Γ_{a-1}, the disk $\mathcal{D}_{1-\varepsilon}(t_j)$ does not overlap with the trace of any path in $\Gamma_{a-1} \setminus \{\gamma_i\}$. Therefore, any path $\gamma \in \Gamma_a \setminus \{\gamma_i\}$ is in the free space of $\mathcal{W}_a$. If γ_i was not blocked in iteration $a - 1$, then $S_a = S_{a-1} \setminus \{s_i\}$, and thus $\Gamma_a \setminus \{\gamma_i\}$ corresponds to a matching of S_a and T_a, which implies that Γ_a exists. Otherwise, $S_a = S_{a-1} \setminus \{s_k\}$, where $s_k \in S_{a-1}$ is the last to ε-block γ_i. Let t_ℓ be the target that

was matched to it in Γ_{a-1}, and let $\gamma_k \in \Gamma_{a-1}$ be the path from s_k to t_ℓ. Observe that $\Gamma_{a-1} \setminus \{\gamma_i, \gamma_k\}$ corresponds to a matching between $S_a \setminus \{s_i, s_k\}$ and $T_{a-1} \setminus \{t_k, t_\ell\}$, and thus to have a matching between S_a and T_a we need to find a path in the free space of $\mathcal{W}_a$ between s_i and t_ℓ. We show that the switch path γ_i' from s_i to t_ℓ as defined in Section 3.3, has this property. Recall that γ_i' is a concatenation of $\gamma[1, w]$, $\overline{\gamma_i(w)s_k}$, and γ_k. The paths γ_i, γ_k are in Γ_{a-1} and therefore lie in the free space of $\mathcal{W}_{a-1}$. By Theorem 5(i), the entire path γ_i' is in the free space of $\mathcal{W}_{a-1}$. We thus need to show that the disk $\mathcal{D}_{1-\varepsilon}(t_j)$ does not overlap with the trace of γ_i'. Indeed, since t_j is an ε-interrupting target, it does not block γ_k, and therefore by Theorem 5(iv) it is also not blocking γ_i'.

The next step is to find an ε-interrupting target t_j. For this, notice that Theorem 4 still holds, because it considers only start and target positions in S_a and T_a. Let $\mathcal{A}$ be the set of robots that are already lying on target positions. Notice that we can also apply Theorem 5 on $\mathcal{W}_a, S_a, T_a$ and Γ_a, and we just need to make sure that no robot in $\mathcal{A}$ can block the switch path γ_k'. As mentioned above, this is true in general, that is, for any path $\gamma \in \Gamma_a$ from $s \in S_a$ to $t \in T_a$, no robot in $\mathcal{A}$ can block γ, because the overlap between any such robot and any point on γ is at most ε. In other words, robots in $\mathcal{A}$ can only interrupt paths in Γ_a.

The final step is using Theorem 7 to construct the clearance paths. Notice that this construction applies for any $p \in S \cup T$, and therefore it can be applied for the ε-interrupted path, even if there were robots in all the start and target positions, except for the endpoints of the path.

Lastly, notice that the concatenation of the paths that correspond to a robot A from all the iterations of the algorithm results in a continuous path. This is because each clearance path starts and ends at the same start/target position. ◀

In the full version of the paper [10] we prove the following theorem.

▶ **Theorem 9.** *Given a polygonal workspace $\mathcal{W}$ with n edges, a set S of m starting positions and a set T of m target positions, there is an algorithm that computes in $\tilde{O}(m^4+n^2m^2)$ time a collision-free motion plan of length $O(OPT)$, where OPT is the length of an optimal solution, or reports that such a motion plan does not exist, assuming that $\rho = \max\{4-2\varepsilon, 2+\varepsilon\}$ and $\omega = \max\{1+\varepsilon, \sqrt{(3-\sqrt{3}-\varepsilon)^2+1}\}$.*

The length of our solution. Note that in fact we bound the size of the solution with the term $(1+c)(\mathrm{OPT}+(4-2\varepsilon)m)$ where c is the maximum number of start/target positions that may collide a unit disk under our robots-separation constraints. For example, when the robots-separation is 2 then $c = 6$. We discuss this further in the full version of the paper [10].

Optimizing the separation bounds. We now optimize the overlap parameter over Constraints 1–5, to achieve either the smallest obstacles-separation or the smallest robots-separation possible. Recall that Constraints 1 and 4 specify the required obstacles-separation bounds, while Constraints 2, 3, and 5 specify the required robots-separation bounds. We highlight three interesting values of ε:

- For $\varepsilon = 0$ the motion plan is monotone, and the length of the solution is at most $\mathrm{OPT}+4m$. In this case, we obtain $\omega = \sqrt{13-6\sqrt{3}} \approx 1.614$ and $\rho = 4$.
- For $\varepsilon > 0$, the motion plan is weakly-monotone.
- For $\varepsilon = \frac{15-6\sqrt{3}}{13} \approx 0.354$, the obstacles-separation is minimized, and we obtain $\omega = 1+\frac{15-6\sqrt{3}}{13} \approx 1.354$ and $\rho = 4-2\frac{15-6\sqrt{3}}{13} \approx 3.291$.
- For $\varepsilon = \frac{2}{3}$, the robots-separation is minimized, and we obtain $\omega = 1\frac{2}{3}$ and $\rho = 2\frac{2}{3}$.

Lower bounds. In the full version of the paper [10] we present lower bounds on the obstacles-separation when the motion plan is required to be (weakly) monotone. We provide an instances of the problem in which (i) $\omega \approx 1.614$ and no monotone motion plan exists, and (ii) $\omega \approx 1.354$ and no weakly-monotone motion plan exists. This shows that to obtain lower obstacles-separation assumptions, a completely different approach is required.

5 The Exodus algorithm: reducing the robots-separation bound

In this section we introduce a different algorithmic strategy for MRMP which we call the *Exodus Algorithm*. This strategy is more global in the sense that it also moves robots that do not directly block a path, and therefore it allows for robots-separation of 2, while requiring obstacles-separation of only 3. Whereas the previous algorithm relies on local path-clearing and larger robots-separation, Exodus replaces these local adjustments with a synchronized outward movement of all robots in order to clear the trace of a geodesic path, so that the selected robot can reach its target via this path. Note that in this section, we assume that the workspace $\mathcal{W}$ is a simple polygon. Due to space considerations, we give an overview of the algorithm and the main result here, see [10] for a formal description and proofs.

Algorithm overview. Given the sets $S, T \subset \mathcal{F}$ as before, the first step of the algorithm is to compute an optimal-assignment path set $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$ for $S, T, \mathcal{W}$. Assume w.l.o.g. that $\gamma_i(0) = s_i$ and $\gamma_i(1) = t_i$ for every $1 \leq i \leq m$. The algorithm then proceeds iteratively. In each iteration, we select a target position $t_i \in T$ which is not yet occupied by a robot. We first extend the path γ_i so that its start and end points are on the boundary of $\mathcal{F}$. The extended path partitions $\mathcal{F}$ into maximally connected sub-regions of $\mathcal{F}$, which we call pockets. We then split each pocket using the angle bisectors corresponding to the reflex subchain of γ_i that defines the pocket. We call the cells in this refined partition Exodus cells. Each Exodus cell contains a single straight-line edge of γ_i. All the other robots are then moved outwards from the extended path by two units (which is possible because $\omega = 3$), and the movement direction for each robot is determined by the Exodus cell in which it lies. This process clears the trace of γ_i, allowing a robot positioned on s_i to move along γ_i until reaching t_i, while all the other robots remain fixed at their offset locations. After the robot reaches its target t_i, all the other robots return to their original positions by applying the inverse movement (which is possible because $\rho = 2$, i.e., no start position is too close to t_i). This iterative "corridor-opening" process motivates the name *Exodus Algorithm*. The procedure repeats until all robots reach their assigned targets. In each iterative step, $m - 1$ robots are moving $4m$ units each, while the selected robot moves along a shortest path to its assigned target (as before, $|\Gamma| \leq \mathrm{OPT}$). Since there are exactly m iterations, we obtain the following theorem.

▶ **Theorem 10.** *Given a simple polygonal workspace $\mathcal{W}$ with n edges, a set S of m starting positions and a set T of m target positions, such that $\rho = 2$ and $\omega = 3$, the Exodus algorithm computes in $\tilde{O}(m^3 + mn^2)$ time a collision-free motion plan of size at most $OPT + 4m^2 - 4m$, where OPT is the length of an optimal solution.*

▶ Remark 11 (Labeled MRMP). In the labeled setting, where each start position is already associated with a specific target, we do not compute an optimal-assignment path set. Instead, we simply compute the geodesic paths between each start and its corresponding target using Wang's Euclidean shortest-path algorithm [17], and then apply the same iterative procedure described above. In this case, the running time becomes $O(m^2 \log n + mn^2)$, as no matching step is required. The motion of the robots during the corridor-opening iterations remains unchanged, and the size of the solution remains at most $\mathrm{OPT} + 4m^2 - 4m$.

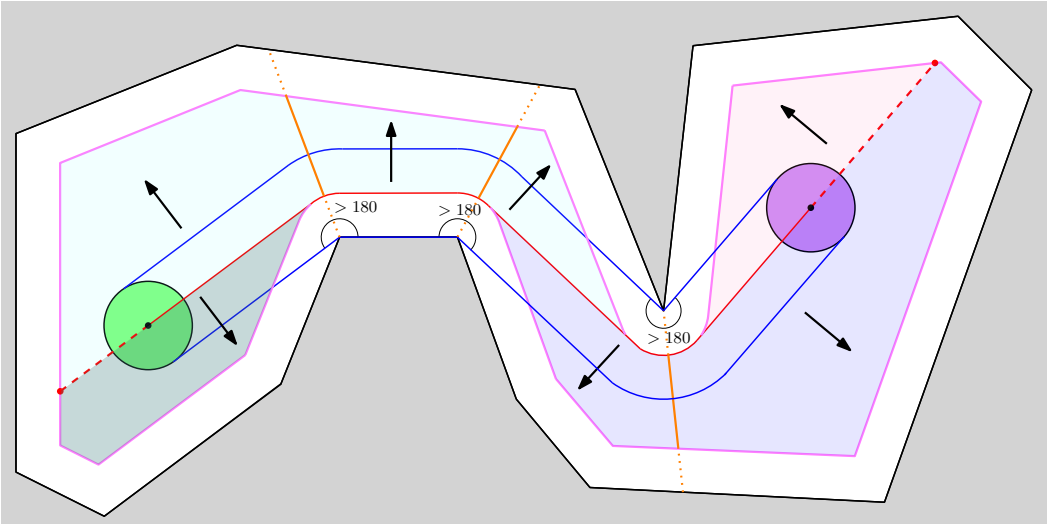


Figure 5 The extended geodesic $\hat{\gamma}$ is in red, and its trace is in blue. In pink is $\partial\mathcal{F}$. Each colored region represents a distinct pocket induced by $\hat{\gamma}$, and the orange rays are the angle bisectors generated at the corresponding reflex vertices of $\partial\mathcal{F}$. These bisectors subdivide each pocket into Exodus cells, within which a single direction vector is illustrated by the black arrows.

6 Discussion and future work

In this paper we provide multi-robot motion plans that require tighter assumptions on the density of the input setting. Note that for the Exodus algorithm we require $\rho = 2$. This separation bound is tight for monochromatic robots-separation, however, the bichromatic robots-separation can potentially be 0. This raises the question of what is the minimum obstacles-separation for which we can remove the robots-separation assumption completely.

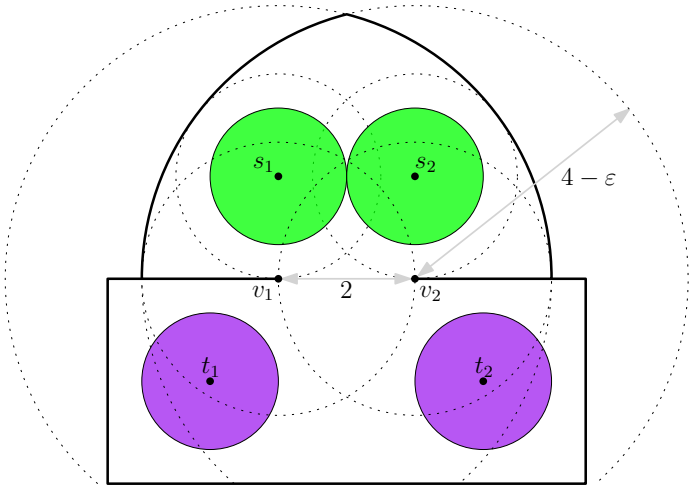


Figure 6 Unlabeled MRMP with obstacles-separation of $1.5 - \varepsilon$, for which no solution exists.

In the full version of the paper [10] we also present lower bounds for the amount of obstacles-separation needed so that a solution to the MRMP problem always exists, without any assumptions on the separation between robots. In the unlabeled case, we prove that obstacles-separation of at least 1.5 is sometimes needed (see Figure 6), and in the labeled case, we observe that obstacles-separation of at least 2 is sometimes necessary (see Figure 7).

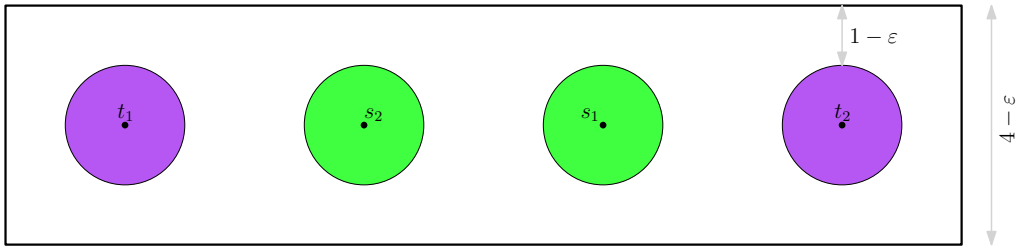


Figure 7 Labeled MRMP with obstacles-separation of $2-\varepsilon$, for which no solution exists.

References

1 Mikkel Abrahamsen, Kevin Buchin, Maike Buchin, Linda Kleist, Maarten Löffler, Lena Schlipf, André Schulz, and Jack Stade. Reconfiguration of unit squares and disks: Pspace-hardness in simple settings. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry, SoCG 2025, Kanazawa, Japan, June 23-27, 2025*, volume 332 of *LIPIcs*, pages 1:1–1:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SOCG.2025.1.

2 Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Transactions on Automation Science and Engineering*, 12(4):1309–1317, 2015. doi:10.1109/TASE.2015.2470096.

3 Pankaj K. Agarwal, Mark de Berg, Benjamin Holmgren, Alex Steiger, and Martijn Struijs. Optimal motion planning for two square robots in a rectilinear environment. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry, SoCG 2025, Kanazawa, Japan, June 23-27, 2025*, volume 332 of *LIPIcs*, pages 5:1–5:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SOCG.2025.5.

4 Pankaj K. Agarwal, Tzvika Geft, Dan Halperin, and Erin Taylor. Multi-robot motion planning for unit discs with revolving areas. *Computational Geometry*, 114:102019, 2023. doi:10.1016/j.comgeo.2023.102019.

5 Pankaj K. Agarwal, Dan Halperin, Micha Sharir, and Alex Steiger. Near-optimal min-sum motion planning for two square robots in a polygonal environment. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4942–4962. SIAM, 2024. doi:10.1137/1.9781611977912.176.

6 Pankaj K. Agarwal, Benjamin Holmgren, and Alex Steiger. Near-optimal min-sum motion planning in a planar polygonal environment. *CoRR*, abs/2510.21639, 2025. doi:10.48550/arXiv.2510.21639.

7 Bahareh Banyassady, Mark de Berg, Karl Bringmann, Kevin Buchin, Henning Fernau, Dan Halperin, Irina Kostitsyna, Yoshio Okamoto, and Stijn Slot. Unlabeled Multi-Robot Motion Planning with Tighter Separation Bounds. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2022.12.

8 Thomas Brocken, G. Wessel van der Heijden, Irina Kostitsyna, Lloyd E. Lo-Wong, and Remco J. A. Surtel. Multi-robot motion planning of k-colored discs is pspace-hard. In Martin Farach-Colton, Giuseppe Prencipe, and Ryuhei Uehara, editors, *10th International Conference on Fun with Algorithms, FUN 2021, Favignana Island, Sicily, Italy, May 30 - June 1, 2021*, volume 157 of *LIPIcs*, pages 15:1–15:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FUN.2021.15.

9 Guillermo Esteban, Dan Halperin, Víctor Ruíz, Vera Sacristán, and Rodrigo I. Silveira. Shortest coordinated motion for square robots. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures - 18th International Symposium, WADS 2023, Montreal, QC, Canada, July 31 - August 2, 2023, Proceedings*, volume 14079 of *Lecture Notes in Computer Science*, pages 430–443. Springer, 2023. doi:10.1007/978-3-031-38906-1_28.

10 Tsuri Farhana, Omrit Filtser, and Shalev Goldshtein. Unlabeled multi-robot motion planning with improved separation trade-offs. *CoRR*, abs/2603.19502, 2026. arXiv:2603.19502.

11 Sándor P. Fekete, Ramin Kosfeld, Peter Kramer, Jonas Neutzner, Christian Rieck, and Christian Scheffer. Coordinated motion planning: Multi-agent path finding in a densely packed, bounded domain. In Julián Mestre and Anthony Wirth, editors, *35th International Symposium on Algorithms and Computation, ISAAC 2024, Sydney, Australia, December 8-11, 2024*, volume 322 of *LIPIcs*, pages 29:1–29:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ISAAC.2024.29.

12 John E Hopcroft, Jacob Theodore Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects; pspace-hardness of the" warehouseman's problem". *The international journal of robotics research*, 3(4):76–88, 1984.

13 Israela Solomon and Dan Halperin. Motion planning for multiple unit-ball robots in $\mathbb{R}^d$. In Marco Morales, Lydia Tapia, Gildardo Sánchez-Ante, and Seth Hutchinson, editors, *Algorithmic Foundations of Robotics XIII, Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics, WAFR 2018, Mérida, Mexico, December 9-11, 2018*, volume 14 of *Springer Proceedings in Advanced Robotics*, pages 799–816. Springer, 2018. doi:10.1007/978-3-030-44051-0_46.

14 Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. In Lydia E. Kavraki, David Hsu, and Jonas Buchli, editors, *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015. doi:10.15607/RSS.2015.XI.046.

15 Kiril Solovey, Jingjin Yu, Or Zamir, and Dan Halperin. Motion planning for unlabeled discs with optimality guarantees. In Lydia E. Kavraki, David Hsu, and Jonas Buchli, editors, *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015. doi:10.15607/RSS.2015.XI.011.

16 Paul G. Spirakis and Chee-Keng Yap. Strong np-hardness of moving many discs. *Inf. Process. Lett.*, 19(1):55–59, 1984. doi:10.1016/0020-0190(84)90130-3.

17 Haitao Wang. A new algorithm for euclidean shortest paths in the plane. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 975–988. ACM, 2021. doi:10.1145/3406325.3451037.

Tilt Automata: Gathering Particles with Uniform External Control

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
L3S Research Center, Hannover, Germany

Jonas Friemel
Department of Electrical Engineering and Computer Science,
Bochum University of Applied Sciences, Germany

Peter Kramer
Department of Computer Science, TU Braunschweig, Germany

Jan-Marc Reinhardt
Department of Electrical Engineering and Computer Science,
Bochum University of Applied Sciences, Germany

Christian Rieck
Institute of Mathematics, University of Kassel, Germany

Christian Scheffer
Department of Electrical Engineering and Computer Science,
Bochum University of Applied Sciences, Germany

Abstract

Motivated by targeted drug delivery, we investigate the gathering of particles in the full tilt model of externally controlled motion planning: A set of particles is located at the tiles of a polyomino with all particles reacting uniformly to an external force by moving as far as possible in one of the four axis-parallel directions until they hit the boundary. The goal is to choose a sequence of directions that moves all particles to a common position. Our results include a polynomial-time algorithm for gathering in a completely filled polyomino as well as hardness reductions for approximating shortest gathering sequences and for determining whether the particles in a partially filled polyomino can be gathered. We pay special attention to the impact of restricted geometry, particularly polyominoes without holes. As a corollary, we make progress on an open question from [Balanza-Martinez et al., SODA 2020] by showing that deciding whether a given position can be occupied remains NP-hard in polyominoes without holes. Our results build on a connection we establish between tilt models and the theory of synchronizing automata.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Uniform control, gathering, full tilt, polyominoes, synchronizing automata

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.44

Related Version *Full Version*: `arXiv:2603.02796` [32]

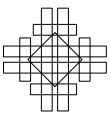
Funding *Sándor P. Fekete, Jonas Friemel, Peter Kramer, Christian Scheffer*: Supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) – project SpaceAnts, 530918134. *Christian Rieck*: Supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) – 522790373.

1 Introduction

We investigate a problem motivated by targeted drug delivery [48]: How can initially dispersed drug-carrying particles be gathered at a common position? Problems of this type are closely related to advances in micro-scale robotics [50]. However, environmental constraints and the



© Sándor P. Fekete, Jonas Friemel, Peter Kramer, Jan-Marc Reinhardt, Christian Rieck, and Christian Scheffer;
licensed under Creative Commons License CC-BY 4.0





severely limited capabilities of robotic agents at micro- and nano-scale rule out autonomous movement. Instead, a global external force, e.g., an electromagnetic field, can be used to move and steer simple robotic particles. The challenge is to use geometry to control a large swarm of particles through uniform signals within a non-uniform environment [12, 16].

Abstract mathematical models [9, 23], commonly called *tilt models* in reference to gravity as a potential external force, have been developed to tackle this problem from an algorithmic perspective. They model the environment as a *polyomino*, i.e., a finite, connected region of the square tiling, with particles positioned at tiles and moving on axis-parallel paths up, down, left, or right. Two major variants exist: In the *single step model* (S1), particles can be precisely controlled via unit steps to adjacent positions, whereas the *full tilt model* (FT) takes imprecision into account by only allowing particles to move maximally until blocked.

Algorithms and complexity results for a variety of applications [15, 25, 44, 51] have been discovered using tilt models. To investigate algorithmic approaches to targeted drug delivery, we consider particles located at the tiles of a polyomino and look for a *gathering sequence*, i.e., a sequence of moves that relocates all particles to the same tile. It is already known that gathering sequences of length $\mathcal{O}(n_c D^2)$ always exist in S1, where n_c is the number of convex corners of the polyomino and D its diameter, but finding sequences of minimum length is NP-hard [13]. Very little is known about gathering in FT, apart from the brief observation in [52] that gathering sequences do not exist for all polyominoes in FT. We remedy this situation with a comprehensive study of gathering in FT.

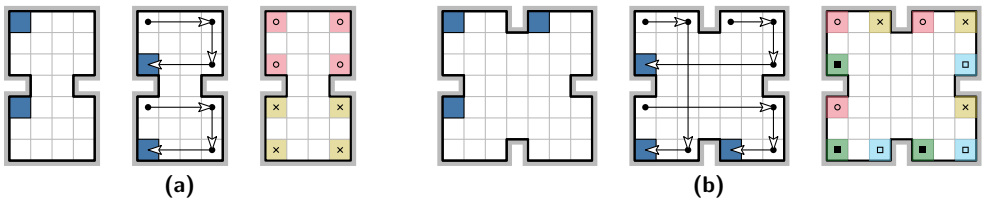


Figure 1 Two examples where gathering a set of particles is impossible in the full tilt model. Both examples depict, from left to right, a set of particles (blue squares) inside a polyomino, a sequence of moves leading to all possible configurations, and a partition of reachable positions yielding a congruence. Example **(a)** prohibits gathering by keeping the particles in different congruence classes, whereas the particles in **(b)** always occupy distinct positions of the same class.

As a first observation, we can see that the full tilt model exhibits several interesting complications that render gathering impossible. The authors of [52] already recognized that particles may become trapped in separate parts of the polyomino, see Figure 1(a). However, this is not the only difficulty. Remarkably, even if all tiles reachable from the initial particle positions are mutually reachable, gathering may still be impossible, see Figure 1(b).

Apart from practical applications, tilt models fascinate with simple, intuitive problems that nevertheless exhibit surprising complexity. A central question is how the computational complexity of a problem relates to the geometric complexity of the environment. Two fundamental problems are Occupancy, which asks if any of a given set of particles can be moved to a specified tile, and ShapeReconfiguration, which asks if a given configuration of particles can be transformed into another given configuration. Both are proven PSPACE-complete in FT for general polyominoes in [7], where the authors pose the complexity for *simple* polyominoes, i.e., polyominoes without holes, as an open problem. We also consider a supplementary class of polyominoes we call *mazes*: collections of horizontal and vertical lines that only intersect in T-shaped and +-shaped junctions, see Figure 2(b).

1.1 Our contributions

We provide a wealth of insights into the gathering of particles in the full tilt model. Let a given polyomino have n corners, $n_c \leq n$ of which are convex. We prove the following results.

- When particles are initially positioned at every tile of a polyomino, a gathering sequence of length $\mathcal{O}(n_c n^2)$ can be computed, if one exists at all (Theorem 4 and Corollary 6).
- There are polyominoes with shortest gathering sequences of length $\Omega(n^2)$ (Theorem 7).
- Finding a shortest gathering sequence is NP-hard, even in simple mazes (Theorem 14).
- There is a polynomial-time 4-approximation for minimizing the length of a gathering sequence in simple mazes (Theorem 16), but there is no polynomial-time constant-factor approximation in general simple polyominoes (unless P = NP; Theorem 17).
- Determining whether a subset of tiles has a gathering sequence is PSPACE-complete in mazes (Corollary 13) and NP-hard in simple polyominoes (Theorem 18).

The last of these results allows us to derive progress on the open question from [7] regarding the complexity of Occupancy and ShapeReconfiguration in simple polyominoes. We show that both remain NP-hard (Corollary 19). Throughout this paper, we utilize a so far overlooked connection between tilt models and automata theory, in particular synchronizing automata. Since this topic is unlikely to be widely known among the computational geometry community, we give a brief survey in Section 3.1. We expect that more insights for tilt models can be gained by exploring this approach further. Conversely, the geometric nature of tilt models could lead to new ways to tackle open questions in automata theory. As we shall see, the discrepancy between the upper and lower bounds on the worst-case lengths of gathering sequences is closely related to a longstanding open problem known as Černý's conjecture.

Full proofs and details for statements marked by ($\star$) as well as several corollaries that exceed the space limit of this extended abstract, including results on the parameterized complexity of tilt problems, can be found in the full version [32].

1.2 Related work

Apart from the aforementioned work [13, 52] on gathering in the single step model, tilt models have been investigated for a variety of purposes. These include realizing permutations of labeled rectangular configurations [9, 10, 11, 73], simulating dual-rail logic circuits [10, 11, 64], sorting and classifying polyominoes [44], mapping an unknown region [51], and filling and draining a polyomino [33]. Notably, gathering all particles in a completely filled polyomino is strictly more difficult than draining to a freely chosen sink: While gathering requires a single position p to be reachable from everywhere, which suffices to guarantee drainability to a sink placed at p [33], this does not ensure gatherability, see Figure 1(b). A major line of research is concerned with the assembly of structures from particles that can be glued together [7, 8, 15, 20, 46, 53, 63], which is strongly related to universal reconfiguration [25].

The study of the computational complexity of fundamental tilt problems was initiated in [9], where the problems now known as Occupancy and ShapeReconfiguration were originally shown NP-hard in FT. They were later settled to be PSPACE-complete [7]. In the single step model, Occupancy is solvable in polynomial time [23], whereas ShapeReconfiguration remains PSPACE-complete [24]. Restricted variants consider moves limited to two or three directions [23, 27] or deterministic move sequences [6].

Taking the maximal movement of the full tilt model and adding individual control leads to the rules of the game Ricochet Robots [30, 36]. Holzer and Schwoon [39] used related ideas in their investigation of Atomix; we cannot reuse their construction because we need a tighter analysis of the length of move sequences. Other related models examine uniform movement in a rectangle, e.g., by sweeping lines [2, 5], or the rules of the game 2048 [1, 3].

The gathering of point particles inside a polygon has been considered by Bose and Shermer [22], but their setting uses repulsion from a point within the polygon, whereas tilt models could be interpreted to use repulsion from a point at infinity. More closely related is the problem of localizing robots with severely limited capabilities [54, 57]. A sequence of movement commands that leads a robot from an initially unknown position in a polygonal environment to a definite position is equivalent to gathering all point particles in the environment at the same final position.

2 Preliminaries

In this section, we give the rigorous definitions omitted from Section 1 in favor of intuition. Every polyomino P has an associated *dual graph* $G_P = (V, E)$, see Figure 2(b), where $V \subset \mathbb{Z}^2$ and $E = \{\{u, v\} : u \in V, v \in V, \|u - v\|_1 = 1\}$, and a *boundary* ∂P, an axis-aligned polygon with integer side lengths that separates V from $\mathbb{Z}^2 \setminus V$. We identify the vertices of V with the tiles of P and call them *pixels*. As a general convention throughout this text, n will denote the number of corners of ∂P, $n_c \leq n$ the number of convex corners, and $N = |V|$ the number of pixels of a given polyomino. Pixels are uniquely determined as the intersection of two *segments*, a *row segment* and a *column segment*, which are maximally contiguous parts of rows and columns; a *corner pixel* is adjacent to two perpendicular sides of ∂P, i.e., next to a convex corner, see Figure 2(a). Formally, a maze is a *thin* polyomino (it contains no 2×2 squares) in which all corner pixels have degree 1, see Figure 2(b).

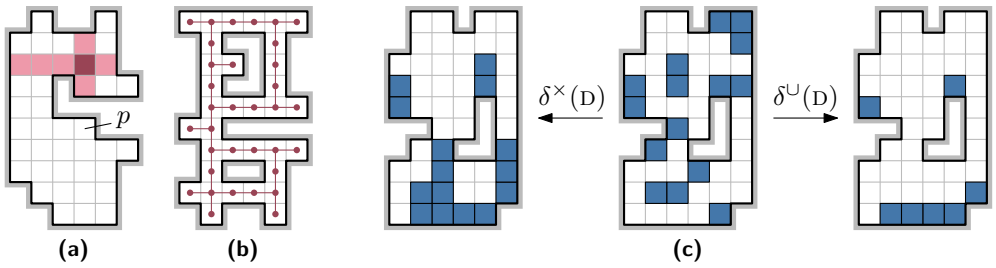


Figure 2 (a) A simple polyomino with a darkly shaded pixel at the intersection of two lightly shaded segments and a corner pixel p. **(b)** A non-simple maze and its dual graph. **(c)** Downwards moves in the blocking and merging variants of the full tilt model acting on a configuration.

A set $C \subseteq V$ of all pixels containing particles is called a *configuration*. Pixels $p \in C$ are *occupied*, those in $V \setminus C$ are *free*. We use the set $\mathbb{D} = \{\text{U}, \text{D}, \text{L}, \text{R}\}$ as shorthand for the directions up, down, left, and right. Formally, a tilt model defines a function $\delta : \mathbb{D} \to (2^V \to 2^V)$, i.e., it maps directions to transformations of configurations. A transformation $\delta(v)$, $v \in \mathbb{D}$, is called a *move*. δ is inductively extended to a function on $\mathbb{D}^*$, the set of (possibly empty) sequences of directions, by setting $\delta(wv) = \delta(v) \circ \delta(w)$, for $v \in \mathbb{D}$ and $w \in \mathbb{D}^*$, and $\delta(\varepsilon) = \mathrm{id}$, where ε is the empty sequence and id the identity function. Particles in the full tilt model are usually blocked when colliding with other particles, but for the purpose of gathering, they merge instead, see Figure 2(c). A formal definition of the *blocking* and *merging* variants for the direction L is as follows; definitions for the other directions are analogous.

Full tilt (blocking): $p \in \delta^{\times}(\text{L})(C)$ if and only if p is one of the $|R \cap C|$ leftmost pixels of the row segment R containing p.

Full tilt (merging): $p \in \delta^{\cup}(\text{L})(C)$ if and only if p is the leftmost pixel of the row segment R containing p and $R \cap C \neq \varnothing$.

Both variants agree on singleton configurations, and the set of singleton configurations is closed under all moves. This gives rise to the function $\delta^1 : \mathbb{D} \to (V \to V)$ defined as $\delta^1(v)(p) = q$ whenever $\delta^\times(v)(\{p\}) = \delta^\cup(v)(\{p\}) = \{q\}$ for $v \in \mathbb{D}$. When δ is clear from the context, we simplify the notation by writing $C \cdot w$ instead of $\delta(w)(C)$ and $C \xrightarrow{w} C'$ to mean $C' = \delta(w)(C)$, and use moves $\delta(v)$ and directions $v \in \mathbb{D}$ interchangeably.

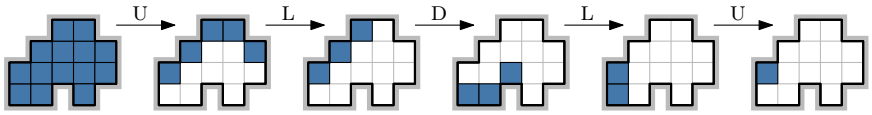


Figure 3 The sequence ULDLU is a gathering sequence for this polyomino in the full tilt model.

A *gathering sequence for a configuration* C is a sequence of directions $w \in \mathbb{D}^*$ such that $|\delta^\cup(w)(C)| = 1$, in which case C is a *gatherable configuration*; a *gathering sequence for a polyomino* P is a gathering sequence for the configuration V, see Figure 3, making P a *gatherable polyomino*. We define $\mathrm{sgs}(C)$ to be the minimum length of a gathering sequence (i.e., the length of a *shortest gathering sequence*) for a gatherable configuration C, and assign $\mathrm{sgs}(P) = \mathrm{sgs}(V)$ for a gatherable polyomino P.

The uniform movement of particles connects the merging variant to the motion of single particles: One can either consider a configuration to be gathered by merging, or move a single particle initially positioned at one of several possible positions aiming to localize it.

▶ **Observation 1.** *For* $w \in \mathbb{D}^*$ *and* $C \subseteq V$, $\delta^\cup(w)(C) = \bigcup_{p \in C} \{\delta^1(w)(p)\} = \bigcup_{p \in C} \delta^\times(w)(\{p\})$.

The single step analogue of Observation 1 is implicitly contained in [13]. Several simple consequences follow: A gathering sequence for a configuration C is also a gathering sequence for every $C' \subseteq C$. If w is a gathering sequence for C, then so is ww' for every $w' \in \mathbb{D}^*$. Prepending works for V: If w is a gathering sequence for V, then so is $w'w$ for every $w' \in \mathbb{D}^*$.

We investigate three problems in the full tilt model for a given polyomino P.

- FullGathering: Find a gathering sequence for P or decide that P is not gatherable.
- ShortestGatheringSequence: Assuming P is gatherable, compute a gathering sequence for P of minimum length $\mathrm{sgs}(P)$.
- SubsetGathering: Decide if a given configuration $C \subseteq V$ is gatherable.

We will use two algebraic properties of the full tilt model. First, every move is idempotent, i.e., $\delta(v) \circ \delta(v) = \delta(v)$ for all $v \in \mathbb{D}$. Secondly, when two moves in opposite directions are applied in sequence, the first is nullified, e.g., $\delta(\mathrm{L}) \circ \delta(\mathrm{R}) = \delta(\mathrm{L})$. Thus, we can generally assume that sequences of moves alternate between perpendicular directions.

3 Gathering particles by synchronizing automata

The notation introduced in Section 2 differs slightly from established conventions and may be considered unnecessarily heavy. It does, however, expedite a point we want to make: Tilt models are transition functions of automata. A *semi-automaton*[1] is a triple (Q, Σ, δ), where Q is a finite, non-empty set of *states*, Σ is a finite, non-empty *alphabet*, and $\delta : \Sigma \to (Q \to Q)$ is a *transition function* mapping letters of Σ to transformations of Q. As usual, δ gets extended to a function over *words* $w \in \Sigma^*$. An *acceptor* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where (Q, Σ, δ) is

[1] All (semi-)automata considered in this text are complete, deterministic, and finite.

a semi-automaton, $q_0 \in Q$ an *initial state*, and $F \subseteq Q$ a set of *accepting states*. The *language accepted by an acceptor* A is the set of words $L(A) = \{w \in \Sigma^* : q_0 \cdot w \in F\}$. For simplicity, we will refer to both semi-automata and acceptors as *automata*. Now it is easy to see that a tilt model defines an automaton $(2^V, \mathbb{D}, \delta)$ for a polyomino P with dual graph $G_P = (V, E)$. We can even restate some established problems in the language of automata theory.

▶ **Example 2.** We are given a polyomino P, configurations $C, C' \subseteq V$, and a pixel $p \in V$.
ShapeReconfiguration: Is $L((2^V, \mathbb{D}, \delta, C, \{C'\}))$ non-empty?
Occupancy: Is $L((2^V, \mathbb{D}, \delta, C, \{C' \subseteq V : p \in C'\}))$ non-empty?

Although interesting, this connection is of no immediate use due to the exponential size of the set of states. Indeed, we would not expect a polynomial description because Occupancy and ShapeReconfiguration are PSPACE-complete in FT [7], whereas emptiness of $L(A)$ can be checked in time polynomial in the size of A [38]. The approach via automata shows its merit when we consider the analogue to gathering sequences, synchronizing words.

3.1 A brief overview of synchronizing automata

A word $w \in \Sigma^*$ is a *synchronizing word* for a set $Q' \subseteq Q$ if there is a state $q \in Q$ such that $p \xrightarrow{w} q$ for all $p \in Q'$, in which case Q' is a *synchronizing set*. The minimum length of a synchronizing word for a synchronizing set Q' is called the *reset threshold* $\mathrm{rt}(Q')$. A *synchronizing automaton* $A = (Q, \Sigma, \delta)$ has a synchronizing set of states Q, in which case we define $\mathrm{rt}(A) = \mathrm{rt}(Q)$. We now briefly summarize some important results; more information can be found in several excellent surveys [38, 60, 69, 70].

It has been (re-)discovered multiple times how to check if an automaton is synchronizing and, if so, find a synchronizing word of length $\mathcal{O}(|Q|^3)$. Eppstein's work [31] is particularly noteworthy for its careful analysis of the time and space requirements. Černý's conjecture states that $\mathrm{rt}(A) \leq (|Q| - 1)^2$ holds for every synchronizing automaton A. It is named in honor of foundational work by Černý, see [29] for a translation into English, that established the corresponding lower bound via a series of synchronizing automata. Remarkably, this seemingly simple conjecture has resisted attempts at a general solution for decades, although it has been proven for special classes of automata, e.g., Eulerian automata [43], monotonic automata [31], and others [70]. Since an improvement of the asymptotic bound $\mathcal{O}(|Q|^3)$ has proven elusive, researchers have looked at the coefficient α of the cubic term. The current best value is $\alpha \leq 0.1654$ due to Shitov [65], who expanded on ideas by Szykuła [67]. Kiefer and Ryzhikov [47] recently improved the running time of an algorithm for the more general problem of finding a word w that minimizes $|Q \cdot w|$ (called the *rank* of w).

A lower bound on the approximation ratio for $\mathrm{rt}(A)$ was improved several times [18, 19, 35], until Gawrychowski and Straszak [34] showed that $\mathrm{rt}(A)$ is NP-hard to approximate within a factor $|Q|^{1-\varepsilon}$ for every $\varepsilon > 0$. On the practical side, engineering efforts have resulted in implementations that find shortest synchronizing words for instances with over 500 states [68].

Deciding if a subset $S \subseteq Q$ is synchronizing is PSPACE-complete, as stated by Natarajan [56] and formally proven by Rystsov [59]. This problem is strongly related to IntersectionNonEmptiness: Given a family of automata over the same alphabet, decide whether they accept a common word. Kozen [49] originally proved this problem PSPACE-complete, and it remains NP-hard for tally automata [55, 66], i.e., automata over a unary alphabet.

3.2 Efficient gathering in the full tilt model

By Observation 1, a gathering sequence for a polyomino P corresponds precisely to a synchronizing word for the automaton $(V, \mathbb{D}, \delta^1)$. Thus, we could solve FullGathering by applying Eppstein's algorithms [31] to $(V, \mathbb{D}, \delta^1)$ to decide if it is synchronizing in $\mathcal{O}(N^2)$ time

and, if so, find a gathering sequence for P in $\mathcal{O}(N^3)$ time. We now sketch algorithms with running times of $\mathcal{O}(n^2)$ and $\mathcal{O}(n_c n^2)$, respectively, that use only ∂P as input. This can make a huge difference as the number of pixels N might not be polynomial in the number of corners n.

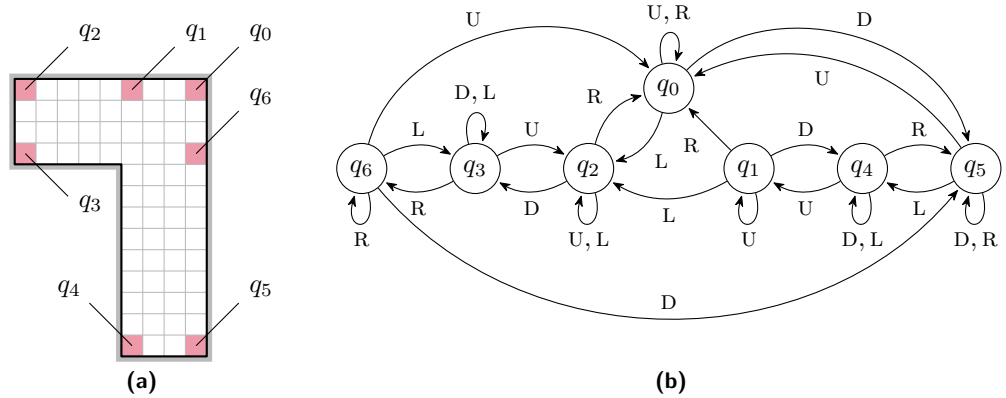


Figure 4 (a) A polyomino P and its significant pixels. **(b)** The full tilt automaton $A(P)$.

The key observation is that $\delta^{\cup}$ very quickly reduces the set of occupied pixels to one of size $\mathcal{O}(n)$. Consider the set $S \subseteq V$ of *significant pixels* that contains all corner pixels and possibly *helper pixels*, which are found by extending sides of ∂P that meet at reflex corners until they hit another side of ∂P. Figure 4(a) illustrates the significant pixels of a polyomino P with helper pixels q_1 and q_6. Observe that $|S| \leq 2n - n_c$ because at most two helper pixels are included per reflex corner. Any sequence of two moves in perpendicular directions changes the position of a particle from any pixel $p \in V$ to one in S, e.g., $\delta^1(\text{DL})(p) \in S$. S is closed under δ^1, i.e., after the move sequence DL we can focus on δ^1 restricted to S. We call the resulting automaton $A(P) = (S, \mathbb{D}, \delta^1)$ the *full tilt automaton* of P, see Figure 4(b).

▶ **Lemma 3** ($\star$). *A polyomino P is gatherable if and only if the full tilt automaton $A(P)$ is synchronizing, in which case* $\text{rt}(A(P)) \leq \text{sgs}(P) \leq \text{rt}(A(P)) + 1$ *holds.*

It is a well-known fact [56, 70] that an automaton (Q, Σ, δ), with $|Q| > 1$, is synchronizing if and only if every unordered pair of states $\{s, t\} \subseteq Q$ has a synchronizing word. This suggests a strategy of iteratively merging pairs. To efficiently determine synchronizing words for pairs in $A(P)$, we construct the *pair automaton* $A(P)^{\leq 2} = (S^{\leq 2}, \mathbb{D}, \delta^{\leq 2})$, where $S^{\leq 2} = \{\{p, q\} : p, q \in S\}$ and $\delta^{\leq 2}(v)(\{p, q\}) = \{\delta^1(v)(p)\} \cup \{\delta^1(v)(q)\}$. Note that $S^{\leq 2}$ also contains all singletons $\{p\} \subseteq S$. A word w with $p \cdot w = q \cdot w$ in $A(P)$ corresponds precisely to a path labeled w from $\{p, q\}$ to $\{p \cdot w\}$ in the digraph underlying $A(P)^{\leq 2}$. Using breadth-first search on this digraph to build a shortest-path forest allows to pre-compute the next step on a shortest path to a singleton for every pair of states, if such a path exists.

To cut down the number of pairs that need to be considered, we initially reduce S to corner pixels only by repeatedly applying two perpendicular moves, e.g., DL. Observe that if $p \in S$ is a helper pixel, then $p \cdot \text{DL} < p$ in terms of their lexicographic order, as at least one coordinate decreases. Thus, $\mathcal{O}(n)$ applications of DL suffice to eliminate all helper pixels (and all corner pixels not at lower-left corners) from S. This way we need to check at most $\mathcal{O}(n_c)$ pairs, improving the running time for non-simple polyominoes with many reflex corners.

▶ **Theorem 4** ($\star$). *Given the boundary ∂P of a polyomino P with $n_c \leq n$ convex corners, a gathering sequence for P, if one exists, can be computed in $\mathcal{O}(n_c n^2)$ time.*

If we are only interested in the (non-)existence of a gathering sequence, we can stop after computing the shortest-path forest. A gathering sequence exists if and only if every pair of states has been assigned a successor on a path to a singleton [31].

▶ **Corollary 5.** *Given the boundary ∂P of a polyomino P with n corners, one can decide in $\mathcal{O}(n^2)$ time whether P is gatherable.*

3.3 Worst-case bounds for gathering sequences in FT

Once a polyomino is known to be gatherable, how long can a shortest gathering sequence be in the worst case? An upper bound matches the running time of the algorithm.

▶ **Corollary 6.** *For every gatherable polyomino P,* $\mathrm{sgs}(P) \in \mathcal{O}(n_c n^2)$.

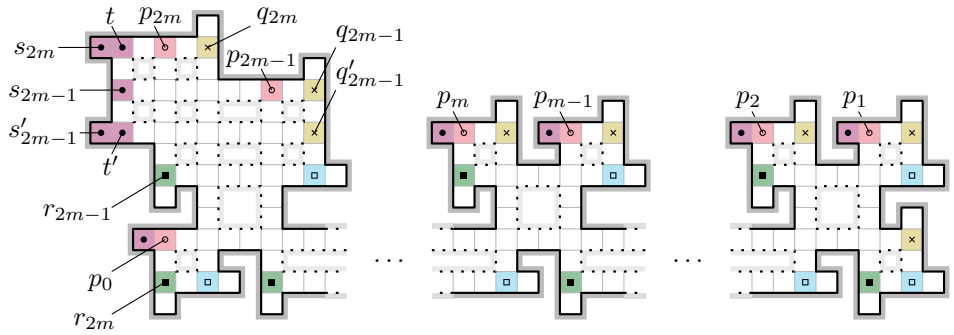


Figure 5 A family of polyominoes P_m with $\mathrm{sgs}(P_m) \in \Omega(n^2)$ in FT. Important classes of pixels are marked with distinct colors and symbols. Dotted areas indicate optional holes that can turn the simple polyomino into a maze, ensuring the construction works for both special cases.

▶ **Theorem 7** (⋆). *There is an infinite family P_m, $m \in \mathbb{N}$, of gatherable simple polyominoes such that* $\mathrm{sgs}(P_m) \in \Omega(n^2)$*, where the number of corners n grows with m.*

Proof sketch. The basic idea of the construction, depicted in Figure 5, is to place two particles on a directed cycle of length $2m + 1$, spaced apart by m positions. Moving against the direction of the cycle leads to dead ends, except at p_{2m-1}. There, LU allows to skip one step on the cycle, which can be used to reduce the distance between the particles by 1. To reduce the distance again, the whole cycle has to be traversed, resulting in $\Omega(m)$ iterations of $\Omega(m)$ moves to gather the particles, for a total of $\Omega(m^2) = \Omega(n^2)$. ◀

4 Simulating binary automata with uniformly controlled particles

If Černý's conjecture is true, then the $\Omega(n^2)$ lower bound on $\mathrm{sgs}(P)$ is actually tight.

▶ **Conjecture 8.** *For every gatherable polyomino P with n corners,* $\mathrm{sgs}(P) \in \mathcal{O}(n^2)$.

The geometric structure of polyominoes should make Conjecture 8 easier to prove than Černý's conjecture. Alas, we must dampen this hope, because particles in polyominoes can be used to simulate binary automata, i.e., automata with alphabet $\Sigma = \{0, 1\}$.

To simulate an automaton $A = (Q, \{0, 1\}, \delta)$, the main idea is to build a polyomino $P(A)$ that has every state $q \in Q$ represented by a pixel $\overline{q}$, and use fixed sequences of moves to embody transitions. A single particle is placed inside $P(A)$, its presence at $\overline{q}$ signifying that

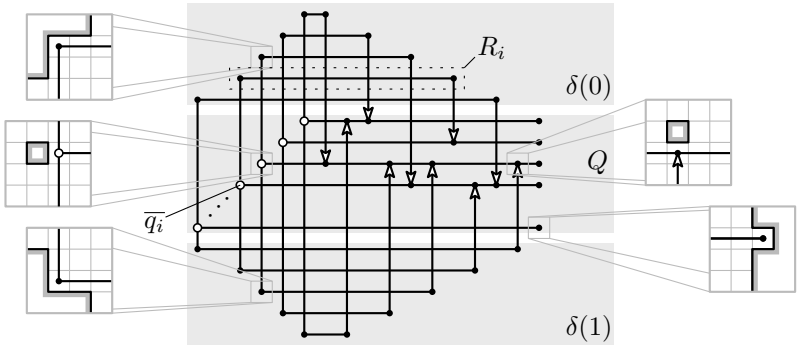


Figure 6 Schematic overview of the construction of $P(A)$ from a binary automaton A. Three blocks represent, from top to bottom, transitions for 0, states $q_i \in Q$, and transitions for 1. We zoom in to show how the schema is realized by the boundary $\partial P(A)$. See Figure 7 for a specific example.

A is in state q. We use the clockwise cycle URDL and the counter-clockwise cycle DRUL to express transitions for 0 and 1, respectively. By carefully constructing the boundary of $P(A)$, we guarantee that all minimal sequences of moves between two pixels $\overline{q_i}$ and $\overline{q_j}$ can be decomposed into combinations of URDL and DRUL, thus ensuring that minimal sequences of moves between pixels representing states are in correspondence with transitions in A.

Without loss of generality, let $Q = \{q_0, q_1, \ldots, q_{|Q|-1}\}$, and set $\overline{Q'} = \{\overline{q} : q \in Q'\}$ for sets $Q' \subseteq Q$. Our construction of $P(A)$ is schematically illustrated in Figure 6. Every state $q_i \in Q$ gets assigned a row of width $6|Q|$ including the representative pixel $\overline{q_i}$. These rows are stacked, with left-aligned rows of width $6|Q|-1$ in between, and the $\overline{q_i}$ are aligned on an upward diagonal, from $\overline{q_{|Q|-1}}$ in the bottom left up to $\overline{q_0}$. A 1×1 hole is placed to the left of every $\overline{q_i}$ except $\overline{q_{|Q|-1}}$, which is next to the outer boundary.

To implement the transitions $\delta(0)(q_i)$ in the top block, rows R_i of width $6i+3$ are stacked and aligned on the left with $\overline{q_i}$. A 1×1 hole is placed below the pixel at the intersection of the rightmost column of R_i and the row of $\overline{q_j}$ when $q_j = \delta(0)(q_i)$. This allows us to simulate a transition for 0 by the sequence of moves URDL. Similar rows and holes are constructed for $\delta(1)(q_i)$ in the bottom block, except now the rows are arranged from bottom to top and of width $6(|Q|-i)-1$, allowing us to simulate 1 by DRUL. The full construction requires $6|Q|$ convex corners and $6(3|Q|-2)$ reflex corners, for a total of $n = 12(2|Q|-1)$, and involves $N \in \mathcal{O}(|Q|^2)$ pixels. Thus, both $\partial P(A)$ and $G_{P(A)}$ can be computed in polynomial time.

For a specific example consider the binary automaton A_0 that accepts the words containing exactly one 0, see Figure 7(a). The polyomino $P(A_0)$ resulting from our construction is depicted in Figure 7(b), where marked pixels show congruence classes of pixels reachable from $\overline{Q}$, in particular those traversed on cycles URDL and DRUL.

▶ **Observation 9.** *For all $p, q \in Q$, $p \xrightarrow{0} q \Leftrightarrow \overline{p} \xrightarrow{\text{URDL}} \overline{q}$ and $p \xrightarrow{1} q \Leftrightarrow \overline{p} \xrightarrow{\text{DRUL}} \overline{q}$.*

▶ **Lemma 10** (⋆). *For all $p \in Q$, with representative pixels $\overline{p} \in P(A)$, and all $w \in \mathbb{D}^*$ with $\overline{p} \cdot w \in \overline{Q}$, there is $w' \in \{\text{URDL}, \text{DRUL}\}^*$ with $|w'| \leq |w|$ that satisfies $\overline{q} \cdot w' = \overline{q} \cdot w$ for all $q \in Q$.*

▶ **Theorem 11.** *A set $S \subseteq Q$ of states of an automaton $A = (Q, \{0,1\}, \delta)$ is synchronizing if and only if its set of representatives $\overline{S}$ is gatherable in $P(A)$, in which case $\operatorname{sgs}(\overline{S}) = 4\operatorname{rt}(S)$.*

Proof. For the forward direction, let $w \in \{0,1\}^*$ be a synchronizing word for S of minimum length. Use Observation 9 to build a gathering sequence $\overline{w} \in \{\text{URDL}, \text{DRUL}\}^*$ for $\overline{S}$ satisfying $|\overline{w}| = 4|w|$. Assume, for sake of contradiction, that there is a gathering sequence $\overline{w}' \in \mathbb{D}^*$

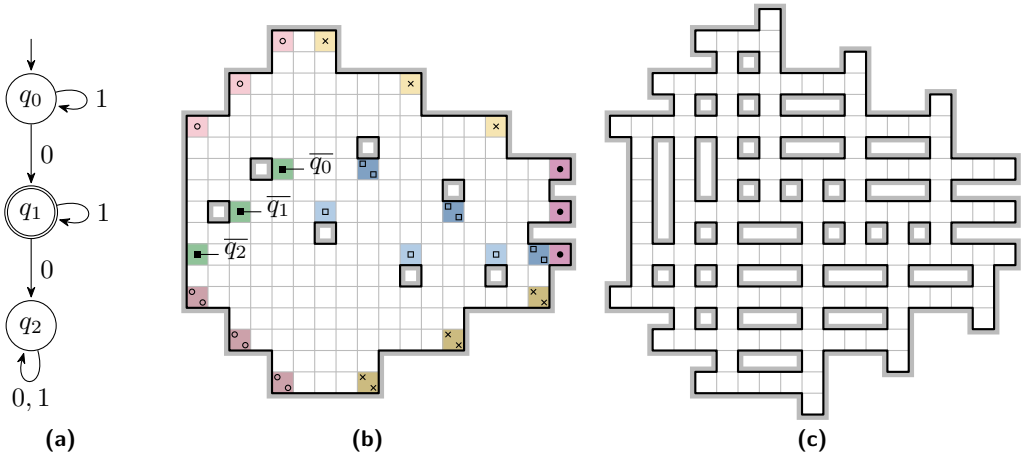


Figure 7 (a) The binary automaton A_0 that accepts all words containing exactly one 0. **(b)** The polyomino $P(A_0)$. Colors and symbols indicate congruence classes of pixels reachable from representatives $\overline{q_i}$. **(c)** $P(A_0)$ turned into a maze $M(A_0)$ by adding holes and pixels of degree 1.

with $|\overline{w}'| < |\overline{w}|$. First apply Lemma 10 to $\overline{w}'$ and then Observation 9 to get a synchronizing word $w' \in \{0,1\}^*$ for S with $|w'| < |w|$, contradicting the assumption that w has minimum length. The backward direction is very similar. ◀

▶ **Corollary 12** (⋆). *If* $\mathrm{sgs}(P) \in \mathcal{O}(n^2)$ *for every gatherable polyomino* P *with* n *corners, then every synchronizing binary automaton* $A = (Q, \{0,1\}, \delta)$ *has* $\mathrm{rt}(A) \in \mathcal{O}(|Q|^2)$.

While a quadratic upper bound on the reset threshold for synchronizing binary automata would not settle Černý's conjecture, it would be remarkable progress. After decades of research, the existence of such an upper bound is still an open problem, suggesting that a proof for Conjecture 8 may be hard to find.

It is easy to adapt the construction of $P(A)$ to produce a maze $M(A)$, at the cost of possibly requiring $\Omega(|Q|^2)$ corners, by adding more holes as well as pixels of degree 1 next to corner pixels, as illustrated for $M(A_0)$ in Figure 7(c).

▶ **Corollary 13.** SubsetGathering *is* PSPACE*-complete, even in mazes.*

Proof. It is easy to see that a non-deterministic algorithm for SubsetGathering could run in polynomial space, which implies containment in NPSPACE = PSPACE [62].

To prove PSPACE-hardness, we reduce from SubsetSynchronizability: Given an automaton $A = (Q, \Sigma, \delta)$ and a set $S \subseteq Q$, determine if S is synchronizing. This problem was originally shown PSPACE-hard by Rystsov [59], and Vorel [72] proved it remains so for $\Sigma = \{0,1\}$. We construct $P(A)$ for a given binary automaton $A = (Q, \{0,1\}, \delta)$ in polynomial time. By Theorem 11, $S \subseteq Q$ is synchronizing if and only if $\overline{S}$ is gatherable. This property remains true when $P(A)$ is turned into a maze $M(A)$. ◀

We want to note that PSPACE-hardness of SubsetGathering in general polyominoes also follows from [7], but we welcome an alternative proof in keeping with our automata theme.

5 Approximating gathering sequences in simple polyominoes

We first show that ShortestGatheringSequence is NP-hard even in simple mazes. To do this, we reduce from ShortestCommonSupersequence where, given an alphabet Σ and a set S of words over Σ, we want to find a shortest word $\overline{w}$ such that for each $w \in S$, there

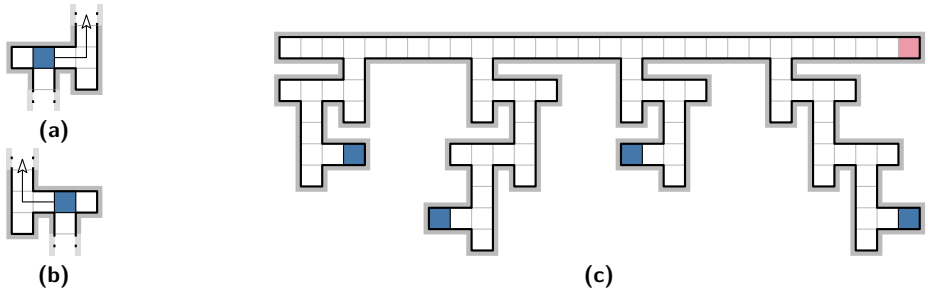


Figure 8 (a) 0-gadget, move sequence RU. **(b)** 1-gadget, move sequence LU. **(c)** The gathering instance for the words 10, 001, 01, and 111. A shortest supersequence is 10011 and the corresponding shortest gathering sequence is LU RU RU LU LU R. The particles gather in the red pixel.

exists a sequence $i_1 < \cdots < i_{|w|}$ with $w_j = \overline{w}_{i_j}$ for each j, i.e., $\overline{w}$ is a supersequence of each word $w \in S$. This problem is already NP-hard if $\Sigma = \{0, 1\}$ [58]. Gerbush and Heeringa [35] used a reduction from a generalized problem to prove inapproximability of the reset threshold of synchronizing automata, which inspired our approach.

▶ **Theorem 14.** ShortestGatheringSequence *is* NP*-hard in simple mazes.*

Proof. Given a set S of words over $\{0, 1\}$, $|S| > 1$, we construct a simple maze as follows: We start with a (sufficiently long) row segment R. For each word $w \in S$, we create a vertical path gadget consisting of the symbol gadgets shown in Figures 8(a) and 8(b) in the order imposed by w. We attach this gadget to the bottom of R, see Figure 8(c). Clearly, this construction is a maze without holes. In each path gadget, we focus on the particle furthest from R as any gathering sequence for these particles is also a gathering sequence for the full polyomino.

We show that there exists a supersequence for S of length k if and only if there exists a gathering sequence of length $2k + 1$ in the corresponding maze. Let $\overline{w} \in \{0, 1\}^*$ be a supersequence for S with $|\overline{w}| = k$. We replace each 0 in $\overline{w}$ with the moves RU and each 1 with LU. In each of the path gadgets, the particle progresses to the next symbol gadget if and only if the symbol in $\overline{w}$ aligns with the moves for its current symbol gadget. Otherwise, it remains in its row. Thus, after the $2k$ moves, all particles have progressed to the row segment R. To gather them in R's rightmost pixel, we add one final R move.

Conversely, let $\overline{w}' \in \mathbb{D}^*$ be a gathering sequence. Without loss of generality, let $|\overline{w}'|$ be minimal. Then, $\overline{w}'$ alternates between horizontal and vertical moves and the first move is horizontal because a vertical move has no effect in the first step. Additionally, $\overline{w}'$ does not contain any D-move as particles can only gather in the row segment R and all shortest sequences to any pixel in R consist of only the moves U, L, and R. Finally, since $|S| > 1$, the last move is horizontal as particles from different path gadgets have to gather in the leftmost or rightmost pixel of R. Therefore, $\overline{w}'$ is a sequence of length $2k + 1$ for some k, consisting of RU and LU subsequences and ending with a horizontal move. We ignore the final move and replace each RU with a 0 and each LU with a 1 to construct a word $\overline{w}$ of length k. As before, a particle progresses to the next symbol gadget exactly if the word in the path gadget contains the corresponding symbol at the position of the particle's gadget. Therefore, since $\overline{w}'$ is a gathering sequence and all particles end up in R, $\overline{w}$ is a supersequence for S. ◀

On the other hand, we can approximate the length of a shortest gathering sequence to within a factor of 4 in simple mazes. To achieve this, we first observe that all such instances have exactly one row or column segment where the particles can gather. Throughout the remainder of this section, we only consider segments consisting of at least two pixels.

▶ **Lemma 15** (⋆). *Let P be a gatherable simple maze with $N > 1$ pixels. Then all row or column segments of P have at least one end point that is a corner pixel and exactly one segment has two corner pixel end points.*

▶ **Theorem 16.** *Let P be a gatherable simple maze. Then we can compute a gathering sequence for P of length at most $4 \cdot \mathrm{sgs}(P)$ in polynomial time.*

Proof. Assume $N > 1$. By Lemma 15, there exists a unique segment R whose end points are both corner pixels and any other segment has exactly one corner pixel. Clearly, all particles have to gather in R as particles initially located in R's corner pixels cannot enter any other segment. It suffices to focus on particles located at corner pixels as all other particles are also gathered by gathering sequences for corner pixels in simple mazes. Now consider a particle p located in the corner pixel of a horizontal segment $S_1 \neq R$. To reach R, p has to pass a sequence of alternating horizontal and vertical segments $S_1, \ldots, S_\ell$ with $S_\ell = R$. Therefore, the shortest sequence moving p to R consists of alternating $\{\mathrm{L}, \mathrm{R}\}$ and $\{\mathrm{U}, \mathrm{D}\}$ moves. Note that p cannot return to S_i after progressing to S_{i+1} and then moving in any direction perpendicular to S_i because the intersection pixel between any consecutive segments S_i, S_{i+1} has degree 3 by the definition of mazes.

Let d be the maximum length of a shortest sequence from any corner pixel to R and consider the sequence URDL. Any particle initially located at a corner pixel progresses to the next segment with at least one of the four moves. Thus, the sequence $(\mathrm{URDL})^d$ is at most four times as long as the shortest sequence moving all particles to R. Finally, the last move in any gathering sequence is perpendicular to R to gather all particles in one of R's end points. ◀

For arbitrary simple polyominoes, there is likely no polynomial-time algorithm approximating $\mathrm{sgs}(P)$ within any constant factor. To show this, we extend our reduction from ShortestCommonSupersequence to unbounded alphabets. If the alphabet size is not fixed, the problem is hard to approximate within any constant factor $\alpha \geq 1$ unless P = NP [41].

We can use the same approach to construct a polyomino as in Theorem 14, i.e., we attach a path gadget consisting of symbol gadgets for each given word to a baseline segment where particles gather. It suffices to generalize the symbol gadget for larger alphabets.

▶ **Theorem 17** (⋆). *Unless* P = NP*, there is no polynomial-time approximation algorithm for* ShortestGatheringSequence *in simple polyominoes with approximation ratio $\alpha \in \mathcal{O}(1)$.*

6 Gathering subsets in simple polyominoes

Like previous reductions for the full tilt model [7, 9], our PSPACE-hardness proof for SubsetGathering in Corollary 13 requires polyominoes with holes. We can retain a weaker hardness result for simple polyominoes by reducing from IntersectionNonEmptiness of *tally automata*, i.e., automata over the unary alphabet $\{0\}$: Given a family $A_i = (Q_i, \{0\}, \delta_i, q_i, F_i)$ of tally automata, determine whether $\bigcap_i L(A_i) \neq \varnothing$, i.e., whether the A_i accept a common word. The NP-hardness of this variant goes back to seminal work by Stockmeyer and Meyer [66]. A newer, independent proof by Morawietz, Rehs, and Weller [55] emphasizes that this variant remains NP-hard if the graphs underlying the automata are cycles. Inspection of the proof reveals that we may also assume the lengths of all cycles to be odd and greater than 1, since they are chosen to be products of large primes. Finally, we may assume all automata to have at least one non-accepting state (or the automaton would accept all words). We implicitly include these restrictions when talking about tally automata in the following.

Given a family $A_i = (Q_i, \{0\}, \delta_i, q_i, F_i)$, $1 \leq i \leq k$, of k tally automata, we construct a simple polyomino $P(A_1, \ldots, A_k)$ and an initial configuration as follows.

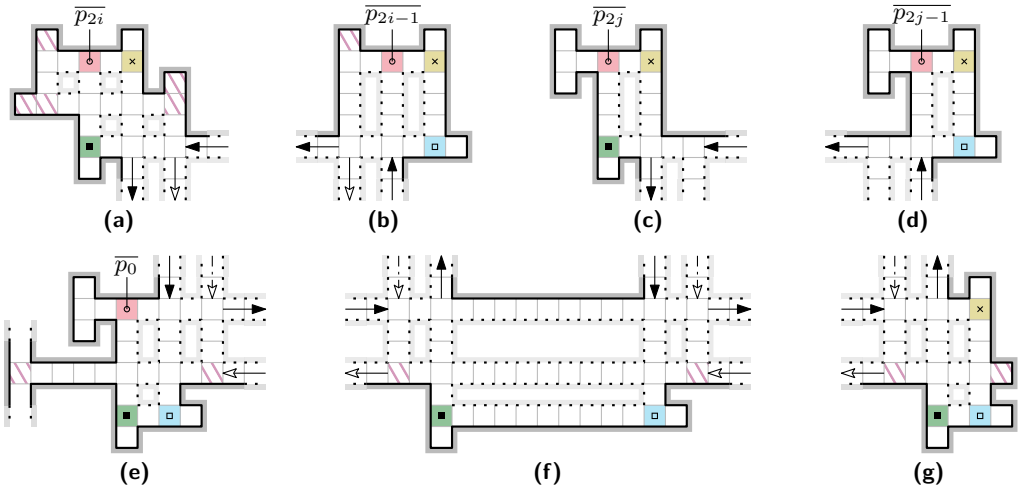


Figure 9 Gadgets used in the reduction from tally automata intersection. Pixels marked with colors and symbols indicate the cycle RDLU between pixels representing states, which is continued between gadgets via the black arrows. The stroked tiling pattern highlights pixels with a safe path to the goal area using the white arrows. Dotted areas specify optional holes that turn the simple polyomino into a maze. **(a)** Accepting state with even index. **(b)** Accepting state with odd index. **(c)** Rejecting state with even index. **(d)** Rejecting state with odd index. **(e)** Start gadget including $\overline{p_0}$ and access to the goal area to the left. **(f)** Bridge gadget. **(g)** End gadget connecting $\overline{p_0}$ to $\overline{p_1}$.

Consider an automaton A_i with $|Q_i| = 2m + 1$ for some $m \geq 1$. Number its states $p_0, \ldots, p_{2m}$ such that $p_0 \notin F_i$ and $p_j \to p_{(j+1) \bmod |Q_i|}$ for all $0 \leq j < |Q_i|$. Place a *start gadget* containing a pixel $\overline{p_0}$ representing p_0, followed horizontally by $m - 1$ *bridge gadgets* and a final *end gadget*; see Figures 9(e)–9(g). Above the partial construction, $2m$ gadgets are placed, from right to left, for the remaining states $p_j \in \{p_1, \ldots, p_{2m}\}$, depending on two factors: the parity of j and whether p_j is accepting. Gadgets for the four possible combinations are depicted in Figures 9(a)–9(d), each containing a representative pixel $\overline{p_j}$.

Partial constructions for A_i are stacked vertically and connected to a *goal area* to the left, which is simply a column segment ending in pixels of degree 1. For a set $S \subseteq \bigcup_i Q_i$, we define $\overline{S} = \{\overline{q} : q \in S\}$ to be the set of representative pixels for states in S. The initial configuration is $C_0 = \bigcup_i \{\overline{q_i}\}$, i.e., the representative pixels for the initial states. Figure 10 depicts a small but complete example of the construction for two tally automata.

▶ **Theorem 18** (⋆). SUBSETGATHERING *is* NP*-hard in simple polyominoes.*

Proof sketch. A common word $0^\ell \in \bigcap_i L(A_i)$ leads to a gathering sequence $(\text{RDLU})^\ell\,\text{LDRDLD}$ for C_0 in $P(A_1, \ldots, A_k)$. For the reverse direction, we argue that if there is any gathering sequence for C_0, then it must first move all particles to representatives of accepting states, which can be done with a (shorter) sequence of the form $(\text{RDLU})^{\ell'}$, which corresponds to a common word $0^{\ell'}$ for the automata. Since the reduction uses a polynomial number of constant-size gadgets, it can be executed in polynomial time. ◀

▶ **Corollary 19.** OCCUPANCY *and* SHAPERECONFIGURATION *are* NP*-hard in the blocking variant of* FT*, even in simple polyominoes.*

Proof. Take the construction from Theorem 18 for k automata and extend the goal area upwards so that there are k pixels above the topmost intersection. The distinction between $\delta^\times$ and $\delta^\cup$ only matters in the goal area, since that is the only segment that can possibly

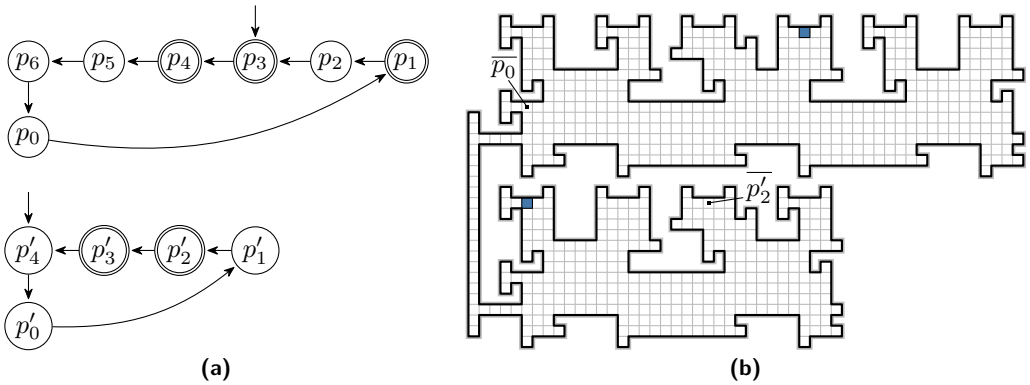


Figure 10 Instance of SubsetGathering produced by the reduction from tally automata intersection. **(a)** Two tally automata, A_1, A_2; both accept the word of length 8. **(b)** The simple polyomino $P(A_1, A_2)$; $(\text{RDLU})^8$ LDRDLD is a gathering sequence for its initial configuration C_0.

Table 1 Overview of complexity results for gathering problems in FT, unless P = NP.

Problem	Geometry	Complexity	Reference
FullGathering	General / Simple	$\mathcal{O}(n_c n^2)$ / $\Omega(n^2)$	**Thms. 4, 7**
		$\mathcal{O}(n^2)$ conj.	**Conj. 8**
ShortestGatheringSequence	Simple	No $\mathcal{O}(1)$-approx.	**Thm. 17**
	Simple maze	NP-hard, 4-approx.	**Thms. 14, 16**
SubsetGathering	Maze	PSPACE-complete	**Cor. 13**
	Simple	NP-hard	**Thm. 18**

contain more than one particle at a time. Now the kth pixel from the top in the goal area can be occupied – and the configuration consisting of the k topmost pixels in the goal are can be reached – if and only if the automata accept a common word. ◀

7 Conclusions

See Table 1 for an overview of our results. The reduction from Section 6 transfers to the related problem of *collecting* [52] a given set of particles into a *connected* configuration in the blocking variant of FT. Our hardness results also apply if the particles are to be gathered at a specified position, since one could simply check all possible positions if this was practical. In the case of FullGathering, it is still possible to find a gathering sequence $w \in \mathbb{D}^*$ such that $V \xrightarrow{w} \{p\}$ for a given pixel p in polynomial time, or decide that no such sequence exists: Assume there are two distinct pixels q_1, q_2 such that $V \xrightarrow{w_i} \{q_i\}$ for $w_1, w_2 \in \mathbb{D}^*$. Then $\{q_1\} \xrightarrow{w_2} \{q_2\}$ and $\{q_2\} \xrightarrow{w_1} \{q_1\}$, i.e., all pixels where particles can be gathered are mutually reachable. Thus, one can check if all particles can be gathered at a given pixel p by using the techniques from Section 3 to find a gathering sequence targeting an unspecified pixel q, and then check if p is reachable from q. If either step fails gathering at p is impossible.

It may be worth reiterating that our algorithmic results are polynomial in the number n of corners of a given polyomino, whereas our hardness reductions produce the dual graph with N pixels, i.e., the respective problems are hard even for a unary representation of space.

A number of open questions remain. While closing the gap between the worst-case bounds $\Omega(n^2)$ and $\mathcal{O}(n_c n^2)$ for $\text{sgs}(P)$ in general polyominoes looks very hard due to its connection to Černý's conjecture, it may be feasible to close the gap in special classes of polyominoes. Is it easier to prove an upper bound $\text{sgs}(P) \in \mathcal{O}(n^2)$ for simple polyominoes?

We have shown SubsetGathering, Occupancy, and ShapeReconfiguration to be NP-hard in simple polyominoes but did not definitively settle their complexity. Can these problems be shown to remain PSPACE-complete in simple polyominoes?

ShortestGatheringSequence can be approximated within a constant factor in simple mazes, which is NP-hard in general simple polyominoes. Is SubsetGathering also easier in simple mazes? We conjecture it to be polynomially solvable in this case.

References

1 Ahmed Abdelkader, Aditya Acharya, and Philip Dasler. 2048 without new tiles is still hard. In *Fun with Algorithms (FUN)*, pages 1:1–1:14, 2016. doi:10.4230/LIPIcs.FUN.2016.1.

2 Hugo A. Akitaya, Greg Aloupis, Maarten Löffler, and Anika Rounds. Trash compaction. In *European Workshop on Computational Geometry (EuroCG)*, pages 107–110, 2016. URL: https://www.eurocg2016.usi.ch/sites/default/files/paper_77.pdf.

3 Hugo A. Akitaya, Erik D. Demaine, Jason S. Ku, Jayson Lynch, and Csaba D. Tóth. 2048 without merging. In *Canadian Conference on Computational Geometry (CCCG)*, pages 285–291, 2020. URL: http://vga.usask.ca/cccg2020/papers/2048%20Without%20Merging.pdf.

4 Hugo A. Akitaya, Maarten Löffler, and Giovanni Viglietta. Pushing blocks by sweeping lines. In *Fun with Algorithms (FUN)*, pages 1:1–1:21, 2022. doi:10.4230/LIPIcs.FUN.2022.1.

5 Hugo A. Akitaya, Maarten Löffler, and Giovanni Viglietta. Pushing blocks by sweeping lines. *Computation in Geometry and Topology*, 2(1):6:1–6:28, 2023. A conference version of this paper appeared at FUN 2022 [4]. doi:10.57717/cgt.v2i1.31.

6 Alberto Avila-Jimenez, David Barreda, Sarah-Laurie Evans, Austin Luchsinger, Aiden Massie, Robert Schweller, Evan Tomai, and Tim Wylie. General computation using slidable tiles with deterministic global forces. In *Innovations in Theoretical Computer Science (ITCS)*, pages 14:1–14:25, 2026. doi:10.4230/LIPIcs.ITCS.2026.14.

7 Jose Balanza-Martinez, Timothy Gomez, David Caballero, Austin Luchsinger, Angel A. Cantu, Rene Reyes, Mauricio Flores, Robert T. Schweller, and Tim Wylie. Hierarchical shape construction and complexity for slidable polyominoes under uniform external forces. In *Symposium on Discrete Algorithms (SODA)*, pages 2625–2641, 2020. doi:10.1137/1.9781611975994.160.

8 Jose Balanza-Martinez, Austin Luchsinger, David Caballero, Rene Reyes, Angel A. Cantu, Robert T. Schweller, Luis Angel Garcia, and Tim Wylie. Full tilt: Universal constructors for general shapes with uniform external forces. In *Symposium on Discrete Algorithms (SODA)*, pages 2689–2708, 2019. doi:10.1137/1.9781611975482.167.

9 Aaron T. Becker, Erik D. Demaine, Sándor P. Fekete, Golnaz Habibi, and James McLurkin. Reconfiguring massive particle swarms with limited, global control. In *Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS)*, pages 51–66, 2013. doi:10.1007/978-3-642-45346-5_5.

10 Aaron T. Becker, Erik D. Demaine, Sándor P. Fekete, Jarrett Lonsford, and Rose Morris-Wright. Particle computation: complexity, algorithms, and logic. *Natural Computing*, 18(1):181–201, 2019. Conference versions related to this paper appeared at ALGOSENSORS 2013 [9], ICRA 2014 [11], and ICRA 2015 [64]. doi:10.1007/S11047-017-9666-6.

11 Aaron T. Becker, Erik D. Demaine, Sándor P. Fekete, and James McLurkin. Particle computation: Designing worlds to control robot swarms with only global signals. In *International Conference on Robotics and Automation (ICRA)*, pages 6751–6756, 2014. doi:10.1109/ICRA.2014.6907856.

12 Aaron T. Becker, Erik D. Demaine, Sándor P. Fekete, Hamed Mohtasham Shad, and Rose Morris-Wright. Tilt: The video – designing worlds to control robot swarms with only global signals. In *International Symposium on Computational Geometry (SoCG)*, pages 16–18, 2015. doi:10.4230/LIPIcs.SOCG.2015.16.

13 Aaron T. Becker, Sándor P. Fekete, Li Huang, Phillip Keldenich, Linda Kleist, Dominik Krupke, Christian Rieck, and Arne Schmidt. Targeted drug delivery: Algorithmic methods for collecting a swarm of particles with uniform, external forces. In *International Conference on Robotics and Automation (ICRA)*, pages 2508–2514, 2020. doi:10.1109/ICRA40945.2020.9196551.

14 Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, Christian Rieck, Christian Scheffer, and Arne Schmidt. Tilt assembly: Algorithms for micro-factories that build objects with uniform external forces. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 11:1–11:13, 2017. doi:10.4230/LIPIcs.ISAAC.2017.11.

15 Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, Christian Rieck, Christian Scheffer, and Arne Schmidt. Tilt assembly: Algorithms for micro-factories that build objects with uniform external forces. *Algorithmica*, 82(2):165–187, 2020. A conference version related to this paper appeared at ISAAC 2017 [14]. doi:10.1007/S00453-018-0483-9.

16 Aaron T. Becker, Golnaz Habibi, Justin Werfel, Michael Rubenstein, and James McLurkin. Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 520–527, 2013. doi:10.1109/IROS.2013.6696401.

17 Mikhail V. Berlinkov. Approximating the minimum length of synchronizing words is hard. In *International Computer Science Symposium in Russia (CSR)*, pages 37–47, 2010. doi:10.1007/978-3-642-13182-0_4.

18 Mikhail V. Berlinkov. Approximating the minimum length of synchronizing words is hard. *Theory of Computing Systems*, 54(2):211–223, 2014. A conference version of this paper appeared at CSR 2010 [17]. doi:10.1007/S00224-013-9511-Y.

19 Mikhail V. Berlinkov. On two algorithmic problems about synchronizing automata (short paper). In *Developments in Language Theory (DLT)*, pages 61–67, 2014. doi:10.1007/978-3-319-09698-8_6.

20 Patrick Blumenberg, Arne Schmidt, and Aaron T. Becker. Computing motion plans for assembling particles with global control. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 7296–7302, 2023. doi:10.1109/IROS55552.2023.10341556.

21 Prosenjit Bose and Thomas C. Shermer. Gathering by repulsion. In *Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 13:1–13:12, 2018. doi:10.4230/LIPIcs.SWAT.2018.13.

22 Prosenjit Bose and Thomas C. Shermer. Gathering by repulsion. *Computational Geometry*, 90:101627, 2020. A conference version of this paper appeared at SWAT 2018 [21]. doi:10.1016/J.COMGEO.2020.101627.

23 David Caballero, Angel A. Cantu, Timothy Gomez, Austin Luchsinger, Robert T. Schweller, and Tim Wylie. Hardness of reconfiguring robot swarms with uniform external control in limited directions. *Journal of Information Processing*, 28:782–790, 2020. doi:10.2197/IPSJJIP.28.782.

24 David Caballero, Angel A. Cantu, Timothy Gomez, Austin Luchsinger, Robert T. Schweller, and Tim Wylie. Relocating units in robot swarms with uniform control signals is PSPACE-complete. In *Canadian Conference on Computational Geometry (CCCG)*, pages 49–55, 2020. URL: https://vga.usask.ca/cccg2020/papers/Relocating%20Units%20in%20Robot%20Swarms%20with%20Uniform%20Control%20Signals.pdf.

25 David Caballero, Angel A. Cantu, Timothy Gomez, Austin Luchsinger, Robert T. Schweller, and Tim Wylie. Fast reconfiguration of robot swarms with uniform control signals. *Natural Computing*, 20(4):659–669, 2021. doi:10.1007/S11047-021-09864-0.

26 David Caballero, Angel A. Cantu, Timothy Gomez, Austin Luchsinger, Robert T. Schweller, and Tim Wylie. Uniform robot relocation is hard in only two directions even without obstacles. In *Unconventional Computation and Natural Computation (UCNC)*, pages 17–31, 2023. doi:10.1007/978-3-031-34034-5_2.

27 David Caballero, Angel A. Cantu, Timothy Gomez, Austin Luchsinger, Robert T. Schweller, and Tim Wylie. Uniform robot relocation is hard in only two directions even without obstacles. *Natural Computing*, 24(1):3–16, 2025. A conference version of this paper appeared at UCNC 2023 [26]. doi:10.1007/S11047-024-10007-4.

28 Ján Černý. Poznámka k homogénnym experimentom s konečnými automatmi. *Matematicko-fyzikálny časopis*, 14(3):208–216, 1964. URL: https://dml.cz/dmlcz/126647.

29 Ján Černý. A note on homogeneous experiments with finite automata. *Journal of Automata, Languages and Combinatorics*, 24(2–4):123–132, 2019. The original article was published in 1964 in Slovak [28]. English translation by Markus Holzer and Bianca Truthe. doi:10.25596/jalc-2019-123.

30 Birgit Engels and Tom Kamphans. Randolphs robot game is NP-hard! *Electronic Notes in Discrete Mathematics*, 25:49–53, 2006. doi:10.1016/J.ENDM.2006.06.062.

31 David Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19(3):500–510, 1990. doi:10.1137/0219033.

32 Sándor P. Fekete, Jonas Friemel, Peter Kramer, Jan-Marc Reinhardt, Christian Rieck, and Christian Scheffer. Tilt automata: Gathering particles with uniform external control. *CoRR*, 2026. arXiv:2603.02796.

33 Sándor P. Fekete, Peter Kramer, Jan-Marc Reinhardt, Christian Rieck, and Christian Scheffer. Drainability and fillability of polyominoes in diverse models of global control. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 74:1–74:19, 2025. doi:10.4230/LIPIcs.ICALP.2025.74.

34 Paweł Gawrychowski and Damian Straszak. Strong inapproximability of the shortest reset word. In *Mathematical Foundations of Computer Science (MFCS)*, pages 243–255, 2015. doi:10.1007/978-3-662-48057-1_19.

35 Michael Gerbush and Brent Heeringa. Approximating minimum reset sequences. In *Conference on Implementation and Application of Automata (CIAA)*, pages 154–162, 2010. doi:10.1007/978-3-642-18098-9_17.

36 Adam Hesterberg and Justin Kopinsky. The parameterized complexity of ricochet robots. *Journal of Information Processing*, 25:716–723, 2017. doi:10.2197/IPSJJIP.25.716.

37 Markus Holzer and Martin Kutrib. Descriptional and computational complexity of finite automata. In *Language and Automata Theory and Applications (LATA)*, pages 23–42, 2009. doi:10.1007/978-3-642-00982-2_3.

38 Markus Holzer and Martin Kutrib. Descriptional and computational complexity of finite automata—A survey. *Information and Computation*, 209(3):456–470, 2011. A conference version of this paper appeared at LATA 2009 [37]. doi:10.1016/J.IC.2010.11.013.

39 Markus Holzer and Stefan Schwoon. Assembling molecules in ATOMIX is hard. *Theoretical Computer Science*, 313(3):447–462, 2004. doi:10.1016/J.TCS.2002.11.002.

40 Tao Jiang and Ming Li. On the approximation of shortest common supersequences and longest common subsequences. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 191–202, 1994. doi:10.1007/3-540-58201-0_68.

41 Tao Jiang and Ming Li. On the approximation of shortest common supersequences and longest common subsequences. *SIAM Journal on Computing*, 24(5):1122–1139, 1995. A conference version of this paper appeared at ICALP 1994 [40]. doi:10.1137/S009753979223842X.

42 Jarkko Kari. Synchronizing finite automata on Eulerian digraphs. In *Mathematical Foundations of Computer Science (MFCS)*, pages 432–438, 2001. doi:10.1007/3-540-44683-4_38.

43 Jarkko Kari. Synchronizing finite automata on Eulerian digraphs. *Theoretical Computer Science*, 295:223–232, 2003. A conference version of this paper appeared at MFCS 2001 [42]. doi:10.1016/S0304-3975(02)00405-X.

44 Phillip Keldenich, Sheryl Manzoor, Li Huang, Dominik Krupke, Arne Schmidt, Sándor P. Fekete, and Aaron T. Becker. On designing 2D discrete workspaces to sort or classify polyominoes. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018. doi:10.1109/IROS.2018.8594150.

45 Jakob Keller, Christian Rieck, Christian Scheffer, and Arne Schmidt. Particle-based assembly using precise global control. In *Algorithms and Data Structures Symposium (WADS)*, pages 513–527, 2021. doi:10.1007/978-3-030-83508-8_37.

46 Jakob Keller, Christian Rieck, Christian Scheffer, and Arne Schmidt. Particle-based assembly using precise global control. *Algorithmica*, 84(10):2871–2897, 2022. A conference version of this paper appeared at WADS 2021 [45]. doi:10.1007/S00453-022-00992-2.

47 Stefan Kiefer and Andrew Ryzhikov. Efficiently computing the minimum rank of a matrix in a monoid of zero-one matrices. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 61:1–61:22, 2025. doi:10.4230/LIPIcs.STACS.2025.61.

48 Matthias Konitzny, Yitong Lu, Julien Leclerc, Sándor P. Fekete, and Aaron T. Becker. Gathering physical particles with a global magnetic field using reinforcement learning. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 10126–10132, 2022. doi:10.1109/IROS47612.2022.9982256.

49 Dexter Kozen. Lower bounds for natural proof systems. In *Symposium on Foundations of Computer Science*, pages 254–266, 1977. doi:10.1109/SFCS.1977.16.

50 Fabian C. Landers, Lukas Hertle, Vitaly Pustovalov, Derick Sivakumaran, Cagatay M. Oral, Oliver Brinkmann, Kirstin Meiners, Pascal Theiler, Valentin Gantenbein, Andrea Veciana, Michael Mattmann, Silas Riss, Simone Gervasoni, Christophe Chautems, Hao Ye, Semih Sevim, Andreas D. Flouris, Josep Puigmartí-Luis, Tiago Sotto Mayor, Pedro Alves, Tessa Lühmann, Xiangzhong Chen, Nicole Ochsenbein, Ueli Moehrlen, Tilman Schubert, Zsolt Kulcsar, Philipp Gruber, Miriam Weisskopf, Quentin Boehler, Salvador Pané, and Bradley J. Nelson. Clinically ready magnetic microrobots for targeted therapies. *Science*, 390(6774):710–715, 2025. doi:10.1126/science.adx1708.

51 Arun Mahadev, Dominik Krupke, Sándor P. Fekete, and Aaron T. Becker. Mapping and coverage with a particle swarm controlled by uniform inputs. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1097–1104, 2017. doi:10.1109/IROS.2017.8202280.

52 Arun V. Mahadev, Dominik Krupke, Jan-Marc Reinhardt, Sándor P. Fekete, and Aaron T. Becker. Collecting a swarm in a grid environment using shared, global inputs. In *International Conference on Automation Science and Engineering (CASE)*, pages 1231–1236, 2016. doi:10.1109/COASE.2016.7743547.

53 Sheryl Manzoor, Samuel Sheckman, Jarrett Lonsford, Hoyeon Kim, Min Jun Kim, and Aaron T. Becker. Parallel self-assembly of polyominoes under uniform control inputs. *Robotics and Automation Letters*, 2(4):2040–2047, 2017. doi:10.1109/LRA.2017.2715402.

54 Rachel A. Moan, Victor M. Baez, Aaron T. Becker, and Jason M. O'Kane. Aggregation and localization of simple robots in curved environments. In *International Conference on Robotics and Automation (ICRA)*, pages 165–171, 2020. doi:10.1109/ICRA40945.2020.9197198.

55 Nils Morawietz, Carolin Rehs, and Mathias Weller. A timecop's work is harder than you think. In *Mathematical Foundations of Computer Science (MFCS)*, pages 71:1–71:14, 2020. doi:10.4230/LIPIcs.MFCS.2020.71.

56 Balas K. Natarajan. An algorithmic approach to the automated design of parts orienters. In *Symposium on Foundations of Computer Science*, pages 132–142, 1986. doi:10.1109/SFCS.1986.5.

57 Jason M. O'Kane and Steven M. LaValle. Localization with limited sensing. *Transactions on Robotics*, 23(4):704–716, 2007. doi:10.1109/TRO.2007.900636.

58 Kari-Jouko Räihä and Esko Ukkonen. The shortest common supersequence problem over binary alphabet is NP-complete. *Theoretical Computer Science*, 16:187–198, 1981. doi:10.1016/0304-3975(81)90075-X.

59 Igor K. Rystsov. Polynomial complete problems in automata theory. *Information Processing Letters*, 16(3):147–151, 1983. doi:10.1016/0020-0190(83)90067-4.

60 Sven Sandberg. Homing and synchronizing sequences. In *Model-Based Testing of Reactive Systems*, pages 5–33, 2004. doi:10.1007/11498490_2.

61 Walter J. Savitch. Deterministic simulation of non-deterministic turing machines (detailed abstract). In *Symposium on Theory of Computing (STOC)*, pages 247–248, 1969. doi:10.1145/800169.805439.

62 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970. A conference version of this paper appeared at STOC 1969 [61]. doi:10.1016/S0022-0000(70)80006-X.

63 Arne Schmidt, Sheryl Manzoor, Li Huang, Aaron T. Becker, and Sándor P. Fekete. Efficient parallel self-assembly under uniform control inputs. *Robotics and Automation Letters*, 3(4):3521–3528, 2018. doi:10.1109/LRA.2018.2853758.

64 Hamed Mohtasham Shad, Rose Morris-Wright, Erik D. Demaine, Sándor P. Fekete, and Aaron T. Becker. Particle computation: Device fan-out and binary memory. In *International Conference on Robotics and Automation (ICRA)*, pages 5384–5389, 2015. doi:10.1109/ICRA.2015.7139951.

65 Yaroslav Shitov. An improvement to a recent upper bound for synchronizing words of finite automata. *Journal of Automata, Languages and Combinatorics*, 24(2-4):367–373, 2019. doi:10.25596/JALC-2019-367.

66 Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In *Symposium on Theory of Computing (STOC)*, pages 1–9, 1973. doi:10.1145/800125.804029.

67 Marek Szykuła. Improving the upper bound on the length of the shortest reset word. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 56:1–56:13, 2018. doi:10.4230/LIPIcs.STACS.2018.56.

68 Marek Szykuła and Adam Zyzik. An improved algorithm for finding the shortest synchronizing words. In *European Symposium on Algorithms (ESA)*, pages 85:1–85:15, 2022. doi:10.4230/LIPIcs.ESA.2022.85.

69 Mikhail V. Volkov. Synchronizing automata and the Černý conjecture. In *Language and Automata Theory and Applications (LATA)*, pages 11–27, 2008. doi:10.1007/978-3-540-88282-4_4.

70 Mikhail V. Volkov. Synchronization of finite automata. *Russian Mathematical Surveys*, 77(5):819–891, 2022. doi:10.4213/rm10005e.

71 Vojtěch Vorel. Subset synchronization of transitive automata. In *Automata and Formal Languages (AFL)*, pages 370–381, 2014. doi:10.4204/EPTCS.151.26.

72 Vojtěch Vorel. Subset synchronization and careful synchronization of binary finite automata. *International Journal of Foundations of Computer Science*, 27(5):557–578, 2016. A conference version of this paper appeared at AFL 2014 [71]. doi:10.1142/S0129054116500167.

73 Yinan Zhang, Xiaolei Chen, Hang Qi, and Devin J. Balkcom. Rearranging agents in a small space using global controls. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3576–3582, 2017. doi:10.1109/IROS.2017.8206202.

Line Segment Visibility in Simple Polygons: Exact, Robust, Scalable Computation and Applications

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
L3S Research Center, Hannover, Germany

Prahlad Narasimhan Kasthurirangan
Department of Applied Mathematics and Statistics, Stony Brook University, NY, USA

Phillip Keldenich
Department of Computer Science, TU Braunschweig, Germany

Fabian Kollhoff
Department of Computer Science, TU Braunschweig, Germany

Chek-Manh Loi
Department of Computer Science, TU Braunschweig, Germany

Michael Perk
Department of Computer Science, TU Braunschweig, Germany

Abstract

The weak visibility polygon of a line segment s inside a simple polygon P, denoted by $V_P(s)$, is the region of the polygon that is visible from at least one point on s. Given its fundamental nature in computational geometry, several algorithms have been proposed to compute weak visibility polygons efficiently, each with different trade-offs in terms of preprocessing time, query time, and space complexity. Although there are many applications that require computing these polygons such as computer graphics, robot motion planning, and network communication systems, there is a lack of *any* implementations of these algorithms in the literature – not to mention one that is exact, robust, and scalable. Furthermore, weak segment visibility polygons are used as basic building blocks in several other algorithms, such as in minimum-link path computation.

In this work, we present an implementation of an optimal linear-time algorithm for computing the weak visibility polygon of a segment inside a triangulated simple polygon. Our implementation provides exact, robust geometric primitives and optimizations to handle large inputs with more than 18 000 000 vertices. We demonstrate two concrete applications: (1) construction of window partitions, a standard data structure in visibility algorithms, and (2) support for optimal minimum-link path queries between two points in a simple polygon, the latter serving as a direct use case of the former. Experimental results on a variety of polygon families confirm that the end-to-end running time scales linearly with the size of the polygon and is dominated by the cost of computing the triangulation, validating the practicality and scalability of the approach. The implementation is released as open source in the format of a CGAL package to support reproducibility and further research.

2012 ACM Subject Classification Theory of computation → Computational geometry; General and reference → Experimentation; Mathematics of computing → Arbitrary-precision arithmetic

Keywords and phrases Visibility, line segments, link distance, window partition, computation, implementation, robustness, scalability, exactness, CGAL

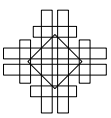
Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.45

Supplementary Material *Software (and Data)*: `https://doi.org/10.5281/zenodo.19194126` [32]

Funding Supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) as part of project *Computational Geometry: Solving Hard Optimization Problems* (CG:SHOP) – 444569951.



© Sándor P. Fekete, Prahlad Narasimhan Kasthurirangan, Phillip Keldenich, Fabian Kollhoff, Chek-Manh Loi, and Michael Perk;
licensed under Creative Commons License CC-BY 4.0





Prahlad Narasimhan Kasthurirangan: Work was carried out during a research stay at TU Braunschweig. Partially supported by the US Office of Naval Research under grant no. N00014-26-1-2088.

Acknowledgements We thank the authors of [4] for sharing parts of their implementation with us.

1 Introduction

Developing efficient methods for geometric problems is at the core of computational geometry. The usual quality measure for such algorithms is their asymptotic worst-case complexity, making non-trivial, linear runtime an ultimate achievement, such as Chazelle's $\mathcal{O}(n)$ triangulation algorithm [11]. While this is undoubtedly one of the intellectual pinnacles of the field, and widely used as a subroutine for achieving fast asymptotic worst-case complexities for many other algorithms, its sophistication also highlights a serious issue: We are not aware of any practical implementation, leaving a serious question posed by none other than Chazelle himself, "Can computational geometry meet the algorithmic needs of practitioners?" [12].

Achieving this goal requires significant additional effort, often involving tough choices and scientific expertise. From the practical side, asymptotic worst-case complexity does not suffice; instead, we require actual *scalability*: what size instances can indeed be solved in reasonable time? This matters not only for hard problems, but also for algorithms for relatively "easy" tasks that are used as repetitive subroutines when dealing with more complex problems.

Another serious challenge is to achieve *robustness*, i.e., ensuring that an implementation does not fail catastrophically in the vicinity of geometric degeneracies? As demonstrated by Kettner, Mehlhorn, Pion, Schirra and Yap [33], even simple algorithms such as Graham's scan [23] can produce completely erroneous results when simply run with floating point arithmetic. An important tool to aid in achieving robustness is the use of *exact* computations, combined with correct handling of all degenerate special cases such as collinear points and overlaps, ensuring that numerical issues cannot lead to any errors. Achieving this with managable performance impact has been one of the goals in community efforts such as CGAL [55], which won the SoCG Test of Time Award in 2023; among many relevant publications, see Boissonat [5] for a practical paper on triangulations. Finally, an implementation can only be used if it is *available*, i.e., its code is published in a usable state and under a suitable license.

In this paper, we demonstrate practical progress along these lines. We describe (and provide) a practical solution for a fundamental problem of Computational Geometry: Compute the *weak visibility polygon of a segment* ℓ within a simple polygon P. This task occurs in a wide variety of problems, such as in robot navigation or optimal surveillance, e.g., in the classical Watchman Problem; see the surveys by Mitchell [45, 44]. In theory, the weak visibility polygon of a segment can be computed in linear time, combining Chazelle's $\mathcal{O}(n)$ triangulation algorithm with an $\mathcal{O}(n)$ algorithm by Guibas, Hershberger, Leven, Sharir and Tarjan [24]. However, to this date, there is no scalable, robust and exact implementation that solves this problem. Indeed, we are unaware of *any* open source implementation. Further relevance is highlighted by the relation to problems such as the Art Gallery Problem, which is not only NP-hard, but even known to be $\exists\mathbb{R}$-complete [1], illustrating the fragility of solutions and the need for robust solutions. Our implementation has all these properties.

- It is *scalable*: We can solve instances with 18 000 000 points within seconds.
- It is *robust* and *exact*: We use exact arithmetic for all computations.
- It is also *available*: it comes as a CGAL package that can be used out of the box.

In this way, we literally follow Chazelle's Recommendation #1 in his 1996 task force report for the first practice track of SoCG, which is also central for SoCG's new practice track: "production and dissemination of geometric code" [12]. In addition, we showcase

two major use cases to demonstrate the practical applicability of our code for large-scale instances: we show how to compute a *window partition* of a polygon P, as well as computing the *minimum link* path between two points in P, for polygons with up to 2 000 000 vertices.

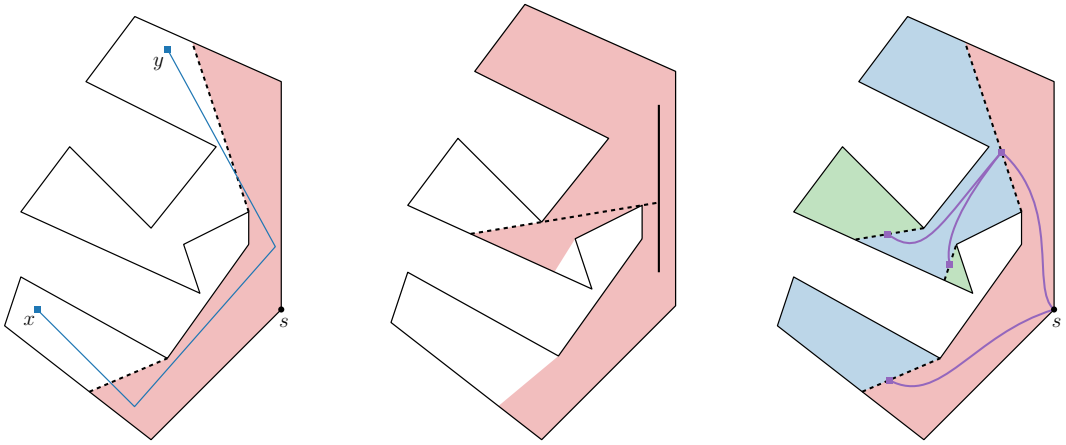


Figure 1 *Left:* $V_P(s)$ is given in red and the windows are marked using dashed lines. A minimum link path between x and y is marked in blue; $d_L(x, y) = 3$. Note that the shortest (Euclidean) path between x and y is not a minimum link path. *Middle:* A segment $\ell \in P$ is marked with a solid black line and $V_P(\ell)$ is marked in red. Note that windows of $V_P(\ell)$ can be defined by points in the interior of ℓ. *Right:* The window partition $\mathcal{W}(s)$ and the window tree T_s of s. Points with link distance one are colored red, with two are colored blue, and with green are colored green. T_s is marked in purple.

1.1 Preliminaries

In the following, P denotes a simple polygon and n its number of vertices. Two points x and y are *visible* to each other if the line segment $xy \subset P$. The *visibility polygon of a point* $s \in P$, denoted by $V_P(s)$, is the set of all points in P visible to s. Given a segment $\ell = pq$ inside of P, the *weak visibility polygon* $V_P(\ell)$ is the set of all points in P that are visible from some point on ℓ, i.e., $V_P(\ell) := \{p \in P \mid \exists\ s \in \ell : p \in V_P(s)\}$; see Figure 1.

For two points $p, q \in P$, the *link distance* $d_L(p, q)$ between them is the minimum number of edges in any polygonal path within P between x and y (see Figure 1, *Left*). Consider a point $s \in P$ and its visibility polygon $V_P(s)$. The vertices of $V_P(s)$ are either vertices of P or the shadows of reflex vertices of P seen by s. The edges of $V_P(s)$ between reflex vertices and their shadows are called *windows*. A window w is a chord of P and divides P into two parts; define $P(w, s)$ to be part of P which does not contain s. The *window partition* $\mathcal{W}_P(s)$ is the partition of P returned by Algorithm 1 described by Suri [50] given input s. We drop the subscript P from $\mathcal{W}_P(s)$ when it is clear from context which polygon we are referring to.

Algorithm 1 Window-Partition(x, P).

1: Compute $V_P(x)$;
2: Let $w_1, w_2, \ldots, w_k$ be the windows of $V_P(x)$;
3: **for** $i = 1$ to k **do**
4: Recursively call Window-Partition$(w_i, P(w_i, x))$;
5: **end for**
6: Output $R(x) := V_P(x)$;

Let $R(w)$ denote the region generated by a window w. The rooted dual tree T_s of $\mathcal{W}(s)$ has as vertices the windows defining $\mathcal{W}(s)$ and edges between two vertices if their regions share a common boundary. The root of T_s is s. We refer to T_s as the *window tree* of s. See Figure 1, *Right*. Let $g(u, v)$ denote the minimum distance of two nodes u, v in T_s and consider a point $p \in R(w)$. The link distance $d_L(s, p)$ is $g(s, w) + 1$ [50].

Constructing a triangulation $\mathcal{T}$ of a polygon P takes linear time [11]. Thus, computing $V_P(\ell)$ for a segment ℓ in P takes linear time – first triangulate P, then use the algorithm from [24]. As we are unaware of any implementation of [11], we use a constrained Delaunay triangulation of P provided by CGAL [5]. This has a worst-case complexity of $\mathcal{O}(n^2)$ [8] but works extremely well in practice, avoids degenerate triangles and thus simplifies our workflow. Our implementation runs in linear time given $\mathcal{T}$ as in [24]. To achieve this runtime, we implemented a specialized data structure called a *finger tree* [30, 29] (see Section 2.1). To the best of our knowledge, all previous implementations of finger trees are in functional programming languages like Haskell or Scala [29, 22]; we used C++ for their iterative versions. We implemented an improvement to [24] suggested in [50]; this allows us to compute $V_P(\ell)$ in time linear in the number of triangles in $\mathcal{T}$ it intersects rather than n. Apart from allowing us to reliably compute $V_P(\ell)$ even in extremely large input polygons, this is crucial to compute window trees in $\mathcal{O}(n)$. All our implementations use exact number types and handle degeneracies. In Section 2, we review the algorithms introduced in [24, 50] and discuss relevant implementation details. We recommend readers familiar with these results to skim through our implementation notes in Section 2 before skipping to Section 3.

1.2 Related work

Computing visibility polygons is central to computational geometry. We focus on weak visibility polygons of segments inside simple polygons; for a more general overview of visibility problems, see [56, 46, 48, 21, 15]. Given a triangulation of a polygon, Guibas, Hershberger, Leven, Sharir and Tarjan [24] presented an $\mathcal{O}(n)$ algorithm that computes the visibility polygon. A number of algorithms have been proposed for data structures that can efficiently compute these for a query segment ℓ, each with its own trade-offs in terms of preprocessing time, query time, and space complexity. We summarize the most relevant ones in Table 1.

Table 1 A summary of theoretical data structures to compute the visibility polygon of a line segment in a triangulated simple polygon. The value k is the size of $V_P(\ell)$ for a query segment ℓ.

Data Structure	Preprocessing Time	Size	Query Time
Bose, Lubiw and Munro [6]	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^3)$	$\mathcal{O}(k \log n)$
Bygi and Ghodsi [9]	$\mathcal{O}(n^3 \log n)$	$\mathcal{O}(n^3)$	$\mathcal{O}(k + \log n)$
Aronov et al. [3]	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(k \log^2 n)$
Chen and Wang [13]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(k \log n)$
Chen and Wang [13]	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(k + \log n)$
Guibas et al. [24]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
With finger tree	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
With CGAL's `Multiset`	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$

Our results are given in bold. We remark that implementing the more recent results with improved output sensitive query time [3, 6, 9, 13] with exact predicates seems extremely challenging. It is also unclear if these will lead to improvements in runtime in practice (see Section 4 for experiments).

Computing segment visibility is a crucial subroutine in several important geometric data structures; perhaps the most well-known one being window partitions and window trees [50]. Other than computing minimum link paths, window partition is also used in convex decomposition algorithms [7, 38] as well as in pursuit evasion problems [52].

Minimum link paths have numerous applications across domains: in motion planning, where turns are expensive and their number should be minimized [47]; in communication systems design under line-of-sight constraints, where a min-link path minimizes the number of necessary repeaters; in placement of telescoping manipulators with flexible telescopic links [34]; in curve simplification by approximating curves with minimum link paths [25, 31, 43]; and in VLSI layouts, where wires typically run along orthogonal axes, motivating the study of link path problems in rectilinear settings [54, 14]. For a comprehensive overview, we refer the reader to [39, Chapter 12] and [40]. There also exist several works on implementing minimum link paths: [53] is restricted to rectilinear paths and is the only work with a publicly available implementation we could find, [20, 57] also consider rectilinear paths, and [17, 58] are restricted to rectilinear environments. Baum, Bläsius, Gemsa, Rutter, and Wegner use minimum link paths in [4] to simplify the boundary of a polygon in a corridor; this is the only other experimental paper on (non-rectilinear) minimum link paths we are aware of. While the authors provided parts of their non-public implementation to us and the paper provides valuable insights into optimizing minimum link path computations for a specific source-target pair, their implementation is tailored to their specific use case and does not offer a method to compute optimal paths between two arbitrary points in a simple polygon.

2 Weak visibility of a segment

In this section, we give an overview of the algorithm by Guibas, Hershberger, Leven, Sharir and Tarjan [24] for computing the visibility polygon of a segment inside a simple polygon P. For a more detailed explanation we refer the reader to the original paper. Consider a triangulated simple polygon P and line segment $\ell = ab$ in P. The algorithm has two steps: first, we compute the *shortest path tree* from both a and b (see Section 2.1 for a definition). Simple data structures can compute this in $\mathcal{O}(n \log n)$ time, reduced to $\mathcal{O}(n)$ using a *finger tree*. In the second step, we compute $V_P(\ell)$ in linear time by iterating over all edges of the polygon and testing if any part of them is visible from ℓ. We only require (amortized) constant time to determine this for each edge of P (and to return the part of this edge that is seen) when we have the shortest path trees of a and b. We discuss details in Section 2.2.

2.1 Shortest path tree

Let P be a simple polygon with n vertices and let s be a given *source vertex* of P. For two points $x, y \in P$, let $\pi(x, y)$ denote the Euclidean shortest path between x and y inside P. Lee and Preparata [37] showed that for all vertices v of P, $\pi(s, v)$ is a polygonal path whose corners are vertices of P and that $\bigcup_v \pi(s, v)$ over all vertices v forms a tree rooted at s. We use $Q_P(s)$ to denote this *shortest-path tree* rooted at s – the vertices of $Q_P(s)$ are vertices of P and (u, v) is an edge of the tree if it is an edge of either $\pi(s, u)$ or $\pi(s, v)$.

Let $\mathcal{T}$ be a triangulation of P and G the dual graph of $\mathcal{T}$. Let $d = uw$ be a diagonal or and edge of P and let a be the deepest common ancestor of u and w in $Q_P(s)$, see Figure 2 *Left*. Then the path $\pi(a, u)$ and $\pi(a, w)$ are two outward convex chains [37], i.e., the convex hull of each subpath lies outside of the region bounded by $\pi(a, u)$, $\pi(a, w)$, and uw. Following [37], we call $F = F_{uw} = \pi(a, u) \cup uw \cup \pi(a, w)$ the *funnel* of d with *cusp* a.

We compute $Q_P(s)$ in linear time by maintaining funnels of diagonals in $\mathcal{T}$ as we process its vertices. We maintain the current funnel F_{uw} as a sorted list $[u_l, u_{l-1}, \ldots, u_1, a, w_1, \ldots, w_k]$ where $a = u_0 = w_0$ is the cusp of F, $\pi(a, u) = [u_0, u_1, \ldots, u_l]$ and $\pi(a, w) = [w_0, w_1, \ldots, w_k]$ with $u_l = u$ and $w_k = w$. We call Algorithm 2 [24] with s and an adjacent vertex v_1 as the initial funnel F; for an interior source point s, we start by locating the triangle containing s and create three funnels, one for each edge, with cusp s.

Algorithm 2 ComputePath(F).

1: $u \leftarrow$ first element of F and $w \leftarrow$ last element of F;
2: $a \leftarrow \text{CUSP}(F)$;
3: $\triangle uwx \leftarrow$ unique unprocessed triangle incident to $d = uw$;
4: Compute $v \in F$ such that vx is a tangent of F at v;
5: Split $F \cup \{x\}$ into $F_1 = [u, \ldots, v, x]$ and $F_2 = [x, v, \ldots, w]$;
6: **if** $v \in \pi(a, u)$ **then**
7: $\quad \text{CUSP}(F_1) \leftarrow v$ and $\text{CUSP}(F_2) \leftarrow a$;
8: **else if** $v \in \pi(a, w)$ **then**
9: $\quad \text{CUSP}(F_1) \leftarrow a$ and $\text{CUSP}(F_2) \leftarrow v$;
10: **end if**
11: $\pi(s, x) \leftarrow \pi(s, x) \cup vx$;
12: Call ComputePath(F_1) and ComputePath(F_2) if ux or wx is a diagonal of P;

We explain Lines 4 and 11 of Algorithm 2 in more detail. As $\pi(a, u)$ and $\pi(a, w)$ are outward convex, the slopes of the segments defining them are monotonic. Thus, we can do a binary search on these slopes to find the point of tangency for x to F in Line 4; see Figure 2, *Left*. We do not store shortest paths explicitly as suggested by Line 11, we simply store back pointers from x to v; $\pi(s, x)$ is computed by following these pointers until we reach s.

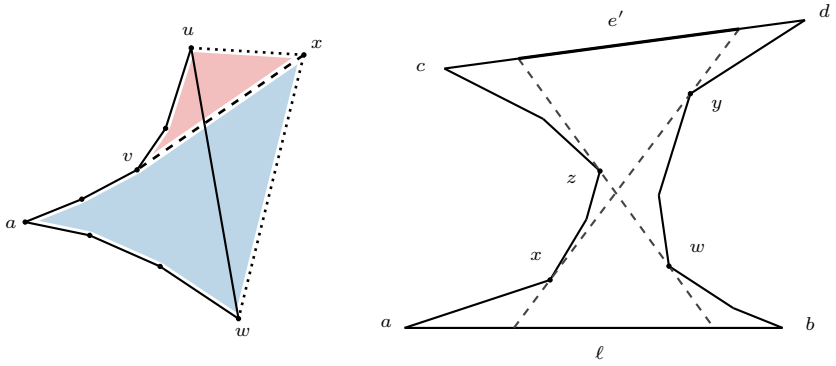


Figure 2 *Left:* Splitting the funnel $F = [u, \ldots, w]$ when a new vertex x is processed. F is split into two sub-funnels $F_1 = [u, \ldots, v, x]$ (red) and $F_2 = [x, v, \ldots, w]$ (blue). *Right:* Hourglass for visibility polygon calculation [24].

Although using binary search as above is straightforward with a binary search tree, it requires $\log n$ time, as $\pi(s, u)$ and $\pi(s, v)$ (and thus our list representing F) might contain $\mathcal{O}(n)$ elements. For an overall linear runtime, we need to find v in (amortized) constant time. This is achieved by storing the list F as a so-called *finger tree* [26]. Our finger trees are modified 2-3 search trees which include pointers (called *fingers*) to the last and first element

of F. They allow for access to v, as well as the splitting of F at v in $\mathcal{O}(\log \delta)$ time, where δ is the distance from v to the nearest finger. See Guibas, McReight, Plass and Roberts [26] for details on properties of finger trees. Gibiansky [22] also provides valuable insights.

Implementation notes. To the best of our knowledge, all implementations of finger trees are in functional programming languages; we implemented an iterative version in C++. As suggested in [24], we can use simpler data structures such as linked lists to represent funnels, resulting in a runtime of $\mathcal{O}(n \log n)$ instead. To compare with our finger tree implementation, we use CGAL's built-in `Multiset` class, as it supports search and split natively. In Section 4, we evaluate the performance of our finger tree implementation against CGAL's `Multiset` class. Collinear points pose a challenge for balanced trees, as orientation checks alone cannot establish a strict ordering. By carefully distinguishing cases, we ensure that the funnel remains free of collinearities at all times.

2.2 Segment visibility polygon

Given a polygon P and a segment $\ell = ab$ inside of P, recall that the weak visibility polygon $V_P(\ell)$ contains all points in P that are visible from a point on ℓ, i.e., $V_P(\ell) = \{s \in P \mid \exists t \in \ell : s \text{ is visible from } t\}$. The following structural lemma is crucial.

▶ **Lemma 1** ([24]). *If a point interior to an edge $e' = cd$ is visible from a segment $\ell = ab$ then (up to swapping c and d) the two paths $\pi(a, c)$ and $\pi(b, d)$ are outward convex.*

Consider such an edge $e' = cd$. The structure formed by ℓ, e', $\pi(a, c)$ and $\pi(b, d)$ is called an *hourglass*, see Figure 2, *Right*. When running the shortest path tree algorithm with a as the source, the funnel $F_{e'}$ has x as its cusp and y as an adjacent vertex. Similarly, when running the algorithm with b as the source, the funnel $F_{e'}$ has w as its cusp and z as an adjacent vertex. These two tangents define the visible section of e'.

During the computation of the $Q_P(s)$, we check, for each vertex v, in constant time if $\pi(s, v)$ is outward convex or not. To extract the visibility polygon, we iterate over the vertices of the polygon and check whether an edge is (partially) visible as explained above. Then we connect continuous visible portions of the boundary, skipping over invisible edges, by connecting visible vertices with their shadow vertices (defined by the tangents). While the algorithm defined in [24] is not output sensitive (taking $\mathcal{O}(n)$ time regardless of the size of $V_P(\ell)$), we explain in Section 2.3 modifications to this algorithm to have this characteristic.

Implementation notes. Although point visibility is available in CGAL [28], our implementation can also compute the visibility polygon of a point s by essentially treating it as degenerate segment, simplifying window extraction. Figure 3 shows the visibility polygon $V_P(s)$ of a query point s. All labeled points are collinear, so the point c is visible from s. We call these low-dimensional features caused by degeneracies *needles* and the polygons that have them *non-regularized*. Often needles are undesired; however, for the implementation of the window partition tree in Section 2.4, we require such visibility polygons. Thus, the visibility polygons we generate are non-regularized; we provide a simple linear-time post-processing function that regularizes visibility polygons for use cases where such polygons are preferred.

2.3 Output sensitivity

A naive implementation of Algorithm 1 using [24] would, for each window, compute a complete shortest path tree of P, resulting in a worst case runtime of $\Omega(n^2)$. Suri [50] suggested modifications to [24] to make the computation of segment visibility output sensitive. The

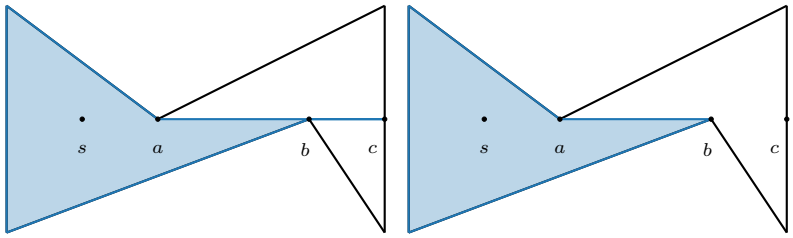


Figure 3 Non-regularized and regularized visibility polygon of s. All labeled points are collinear.

main idea is to synchronize the shortest path tree computation of both endpoints of a segment ℓ and abort the computation of the visibility polygon if an edge is no longer visible from ℓ.

Let P be a simple polygon and $\mathcal{T}$ be a triangulation of the interior of P. For a given segment $\ell = ab$, we start by computing the shortest path tree rooted in a and b along the triangles intersected by ℓ. When processing the next point x in a triangle $\triangle uwx$, we only continue with a diagonal ux if either $\pi(a, u)$ and $\pi(b, x)$, or $\pi(a, x)$ and $\pi(b, u)$ are both outward convex. Symmetrically, we only continue with diagonal wx if either $\pi(a, w)$ and $\pi(b, x)$, or $\pi(a, x)$ and $\pi(b, w)$ are both outward convex. The resulting runtime to compute a visibility polygon $V_P(\ell)$, is $\mathcal{O}(k_\ell)$, where k_ℓ is the number of triangles intersected by $V_P(\ell)$. To ensure that the implementation achieves this runtime, we carefully adjust the traversal of the polygon boundary to only consider vertices that are part of the considered triangles.

Note that k_ℓ can be arbitrarily large compared to the size of the visibility polygon $V_P(\ell)$, e.g., for a triangular visibility polygon that intersects many triangles of $\mathcal{T}$. However, exploiting the fact that each triangle is intersected by at most three partitions [50], we obtain a linear-time algorithm to compute the window partition $\mathcal{W}(s)$ of a point s in P. Note that to start the shortest-path tree computation, we have to locate a triangle intersecting ℓ. We do this using CGAL's built-in `locate` function, which in the worst case has a linear runtime.

2.4 Window partition

To compute the window partition $\mathcal{W}(s)$, we combine Algorithm 1 with our output sensitive visibility polygon computation. The algorithm initially computes a single triangulation $\mathcal{T}$ of P, which we use to compute the non-regularized visibility polygon of s, giving us the initial set of windows. Then, for each window w, we can compute the visibility polygon $R(w) := V_{P(w,s)}(w)$ on the remaining polygon $P(w, s)$. We reuse $\mathcal{T}$ by checking if the next vertex is in $P(w, s)$. This window partition forms the basis for our implementation of the minimum link path algorithms in Section 3.

3 Minimum link path

For a simple polygon P and two interior points s and t, the *link distance* $d_L(s, t)$ between s and t is the minimum number of line segments (links) required to form a polygonal path from s to t that is completely contained in P. The link metric has received considerable attention in computational geometry [49, 50, 42, 2, 41, 35, 27], typically focusing on answering distance queries and not constructing actual minimum link paths. Using window partition trees and other observations, Arkin, Mitchell and Suri [2] proposed an algorithm that can answer link distance queries in $\mathcal{O}(\log n)$ time with $\mathcal{O}(n^3)$ preprocessing time and space. Although the authors mention that the path can be constructed by backtracking through the windows, no details are provided. We implemented two linear time algorithms [50] for answering minimum link path queries based on window partitions: an exact algorithm and an (additive) approximation algorithm that is at most four links longer than the optimal solution.

3.1 Exact algorithm

To construct exact minimum link paths from a source point s to any target point t in a simple polygon P, we use the window partition $\mathcal{W}(s)$ rooted at s. First, we locate the region $R(w)$ that contains t and then traverse the unique path $(w, w_1 \dots s)$ from w to s in T_s. Our construction of $\mathcal{W}(s)$ informs us of a minimum link path from s to t in P that has a vertex on each of the windows $w, w_1, \dots, s$ in that order. Constructing T_s takes $\mathcal{O}(n)$ time and space [50] and locating the target node takes $\mathcal{O}(n)$. Point location could be reduced to $\mathcal{O}(\log n)$ time with preprocessing, but the preprocessing takes super-linear time [15, Chapter 6]. In Section 3.3, we explain how we reconstruct the minimum link path in P in linear time.

3.2 Approximation algorithm

To compute an additive $(\mathrm{OPT}+4)$ approximative minimum link path from p to q in a simple polygon P, we can use the window partition $\mathcal{W}(s)$ of an arbitrary point $s \in P$. For two vertices u and v of T_s, let $g(u, v)$ denote the distance in T_s between them.

▶ **Lemma 2** ([50]). *Let w_1, w_2 be two nodes of T_s, and let $p \in R(w_1), q \in R(w_2)$ be two points of P. Then $g(w_1, w_2) - 1 \leq d_L(p, q) \leq g(w_1, w_2) + 3$.*

The idea is to first identify the two regions $R(w_p)$ and $R(w_q)$ that contain p and q in $\mathcal{W}(x)$ respectively. Then, we identify the lowest common ancestor w_0 of w_p and w_q in T_s. This allows us to construct two paths: one from w_p to w_0 and one from w_q to w_0. Finally, we have to connect the two paths with an additional link on w_0.

Note that, even if $w_0 = w_p$ or $w_0 = w_q$, i.e., the path is directed in T_s, we may still require an additional link using the above algorithm: let $w_0 = w_p$ and $w_0, w_1, \dots, w_q$ be a directed path in T_s. As in Section 3.1, we construct a path from q to p with one vertex v_i on each window w_i for $0 \leq i \leq t$. Unlike the exact algorithm, we cannot guarantee that p is visible from v_0, we can only guarantee the existence of a point v_0' on w_0 that is visible from p. Therefore, we may need to add an additional link from v_0 to v_0' and then to p.

We precompute window partitions rooted at a set of seed vertices and answer queries by probing a small subset of them. At query time, a small set of candidate seed vertices is chosen, e.g., the k nearest to the endpoints. Additionally, we post-process the candidates by removing any introduced loops. The algorithm returns a path with a minimum number of links among all candidates and runs in time proportional to the number of candidates and the cost of locating nodes and reconstructing paths, i.e., $\mathcal{O}(k \cdot n)$ in the worst case for k candidates. Note that Lemma 2 also provides a lower bound on the link distance between p and q that can be used to evaluate the quality of the approximated path.

3.3 Path reconstruction

Now we reconstruct the path from s to t for the exact algorithm detailed in the prior sections. Specifically, we look to construct $(s, p_0, p_1, \dots, p_j, t)$, where p_i belongs to the window w_i. This is done backwards from t. For the step from w_0 to s, any point on w_0 works because s sees all points on w_0. For going from a window w_{i+1} to w_i, this can easily be done in $O(1)$ by extending w_{i+1} to a line and determining its intersection with w_i. However, especially if t is not fixed when constructing the window partition, e.g., because we are answering multiple queries with the same source s, we need to find a point on the last window w_j that sees t. There is a straightforward algorithm that can do this in linear time: compute $V_{R(w_j)}(t)$ and intersect this with w_j to obtain the maximal connected subsegment of w_j that sees t. In practice, this approach is not preferable due to the lack of efficient implementations of

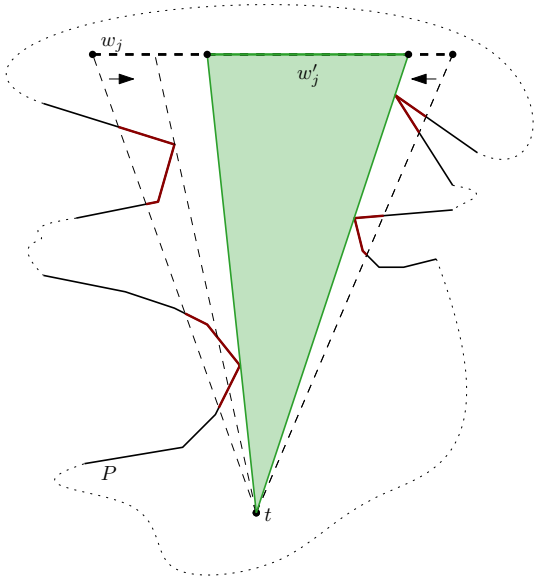


Figure 4 Illustration of the routine described in Section 3.3. The interior point t emits two initial rays defining a cone through the endpoints of window w_j. Several boundary chains (in red) intersect these rays; walking these chains tightens the cone whenever an occluding vertex yields a stricter supporting direction. The intersection of the final cone with w_j is the visible subsegment w'_j. The arrows indicate the direction in which the cone tightens.

visibility algorithms for non-regularized polygons that often appear in the generated window partitions, see Figure 3. The linear-time algorithm available in CGAL [28] is not directly applicable, as it requires the polygon to be simple. Additionally, the triangular expansion approach implemented in CGAL, which is often preferred in practical applications, requires a triangulation of each window, which cannot be directly derived from our triangulation of P, as windows are not necessarily edges of the triangulation; even if we had such a triangulation, the interface of CGAL's visibility package does not allow specifying a triangulation as input. Even without these issues, triangulating each region separately would result in a (practical) superlinear worst-case runtime.

To avoid these problems, we compute the maximal connected subsegment w'_j of w_j that is visible from t in linear (in the complexity of $R(w_j)$) time as follows. Consider the ray between t and the left endpoint of w_j; this possibly intersects $\partial R(w_j)$. Consecutive hitting points define chains of edges on one side of the ray that occlude parts of w. March through the vertices in these chains and choose the vertex defining the rightmost ray from t through it. We find the supporting ray for the right endpoint symmetrically. If these supporting rays cross, w_j is not visible from t (an impossibility). Otherwise, they define the maximal subsegment on w_j visible from t; see Figure 4 for an illustration. Because each other step of path reconstruction takes time $O(1)$, this allows path reconstruction for source s in time $O(|R(w_j)| + j)$ once $\mathcal{W}(s)$ is computed and the window containing t is identified.

For the approximation algorithm, we apply the above procedure for both the windows containing s and t; apart from this and an added segment on the window defining their deepest common ancestor, the situation is analogous.

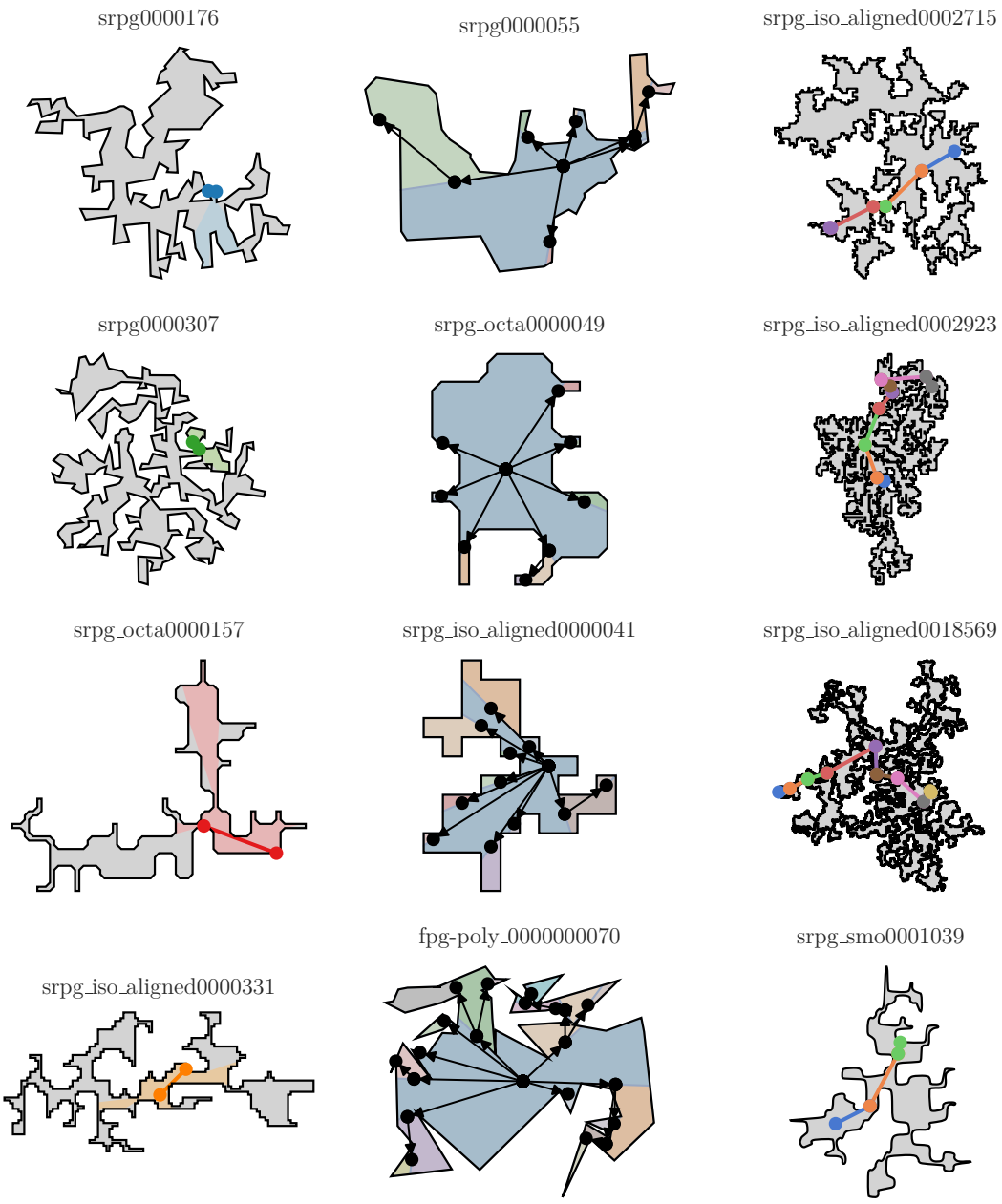


Figure 5 Outputs for the algorithms described in our paper. *Left Column:* Weak visibility polygon of a segment in a simple polygon. *Middle Column:* Window partition of a simple polygon from a source point s. *Right Column:* Minimum link path between two points s and t in a simple polygon.

4 Experimental evaluation

In this section, we present the experimental evaluation of our implementations, see Figure 5 for some example outputs. All experiments were run on otherwise idle workstations equipped with AMD Ryzen 7 5800X CPUs and 128 GiB of DDR4-3200 memory, using Ubuntu Linux 24.04.3 LTS as operating system. The core routines were implemented in C++17, compiled against CGAL 6.0.1 and Boost 1.83.0 using g++ 13.3.0. Our instances and code are publicly available[1]. To guard against programming errors, our code contains large amounts of optional checks and assertions. We also ran a suite of tests with these checks enabled, using instrumentation with tools such as the undefined behavior and address sanitizers of our compiler, which should catch essentially all memory errors. We validated that all paths and visibility polygons found in our experiment stay within the given polygon, and any approximate path has at least the same number of links as the corresponding exact link path.

4.1 Weak line segment visibility

For evaluating our implementation, we consider the following research questions.

RQ1 How does our implementation scale to large inputs, and how is the total runtime distributed onto the individual subroutines?

RQ2 Is there a measurable difference between using finger trees and CGAL's multiset? Which of the two approaches is faster?

RQ3 How does the performance of our implementation depend on the size of the output?

To address these questions, we randomly generate segments for simple polygons from the Salzburg Database of Geometric Inputs[2] [19], the largest of which has 18 556 888 vertices. These polygons are of various types and include polygons with many collinear points or other degeneracies. To generate segments, we sample a source point uniformly at random within the given polygon, compute the visibility polygon of that point, and then either sample a target point in the visibility polygon or pick a random vertex of it. The former approach corresponds to sampling segments inside polygons uniformly at random, whereas the latter focuses on placing one of the endpoints on the boundary of the polygon, exercising edge cases. Using each polygon and each approach twice results in a total of 24 096 instances. For each such instance, and each of the two shortest path tree data structures, we compute the exact visibility polygon five times, measuring the runtime (both total and of various subroutines). Figure 6 depicts the resulting runtimes. We can see that the overall runtime is dominated by computing a triangulation of the input polygon, which we do via CGAL's constrained Delaunay triangulation (CDT). Even CGAL's `mark_in_domain` routine, which simply flags the CDT's triangles that constitute the polygon, is typically more expensive than the actual visibility computation. The fact that the runtime for the visibility computation seems to be almost unaffected by the size of the input polygon can be explained by the fact that it – except for a few small operations such as finding the segment endpoints in the triangulation – mostly depends on the size of the output polygon; see Figure 7, *Left*. On our inputs, the implementation using finger trees is slightly faster than the one using CGAL's `Multiset`, see Figure 7 *Right*. In a preliminary microbenchmark based on building funnels by left insertion, our finger trees became significantly faster already at around 15 funnel vertices; this makes a difference even for the typically small funnels we encounter in our algorithm. We therefore used our finger trees in our other experiments.

[1] `https://doi.org/10.5281/zenodo.19194126`

[2] We used all valid simple polygons from `https://zenodo.org/records/3784789` not labeled as contrived.

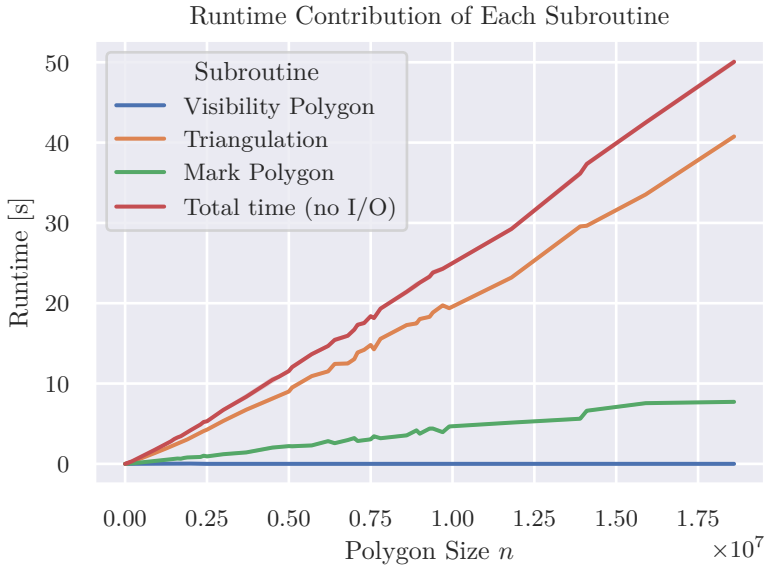


Figure 6 The runtime of different subroutines (triangulating the polygon, marking faces in the polygon, and computing the visibility polygon itself).

4.2 Minimum link paths

Recall that computing a window partition is a basic routine that is useful in different contexts. Furthermore, we only need one window partition from a source point s to compute shortest link-distance paths to any target t, and only a constant number to approximate shortest link-distance paths for any source-target pair inside our polygon. We therefore study the performance of our window partition on its own, as well as in the context of minimum link paths, addressing the following research questions.

RQ4 How does our window partition implementation scale to large polygons?

RQ5 How does it compare to the other parts of our exact minimum link path implementation?

RQ6 How does the approximate minimum link path algorithm perform w.r.t. runtime and solution quality?

To answer these questions, we randomly selected a sample of 1000 simple polygons with at most 2 000 000 vertices from [19]. For each such polygon P, we generate 10 source points $s \in P$ uniformly at random and computed a window partition for each source. Furthermore, for each source point s, we generate a random target point $t \in P$ outside of the visibility polygon of s and computed an exact minimum link s-t-path, as well as an approximate minimum link path using $k = 7$ randomly chosen source points for the window partitions in our approximation. The resulting runtimes are depicted in Figure 8. For each path computed using the approximation algorithm, Figure 9 shows the absolute, additive error depending on the number of window partitions k; the quality of the approximation notably improves with higher k. One should, however, note that increasing k makes both preprocessing and path reconstruction more expensive, both scaling linearly in k.

Overall, we find that both our window partition and minimum link path implementation scale well to large polygons; in fact, in our experiments, time spent inputting, verifying and outputting polygons was typically orders of magnitude higher than pure computation time. If we want to query minimum link paths from a single source s, each subsequent query helps

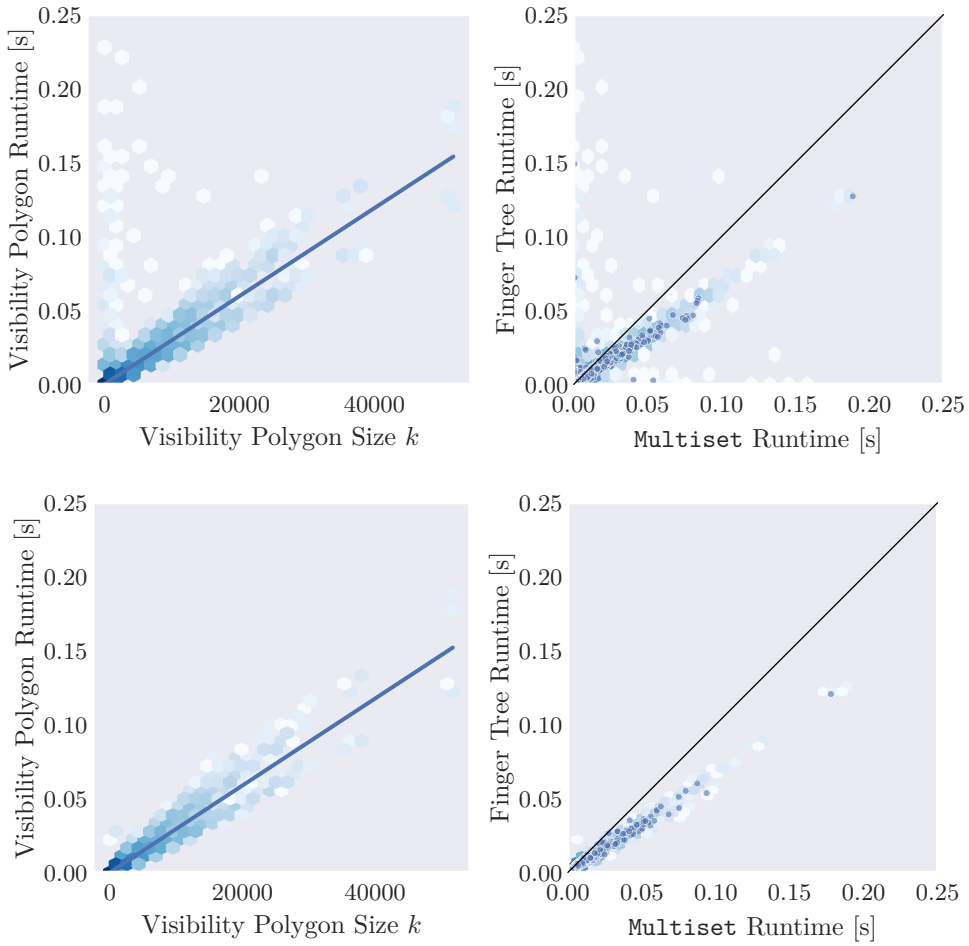


Figure 7 *Top Left:* Runtime dependence on the size of the output visibility polygon, excluding preprocessing such as triangulating the polygon. The hexagonal bin plot shows the underlying distribution similar to a scatter plot, for which we have too many data points; darker hues correspond to a higher concentration of data points, the gray area contains none. *Top Right:* Plot of the runtime using CGAL's `Multiset` and our finger tree implementation with a random sample of data points; the finger tree is better for points below the black diagonal. *Bottom:* The same data as the top row, but dropping the two worst runtime measurements (out of five) for each instance/data structure combination. As our algorithm is deterministic, it is likely that the noise in the top row is simply due to runtime artifacts such as interrupts and scheduling of background processes.

amortizing the cost of computing a window partition; similarly, for doing many queries for different sources and targets in the same polygon, the approximate algorithm can be used to amortize the cost of computing a constant number of window partitions. Depending on the relative importance of runtime versus solution quality, the approximation may be preferable.

5 Conclusions

We presented a scalable, robust and exact implementation for computing the weak visibility polygon of a line segment in a simple polygon. We also demonstrated two concrete applications: computing the window partitioning, and computing the link distance between two points inside a simple polygon. For further improvement on the algorithm engineering side, we could parallelize the shortest path tree and window partition implementations.

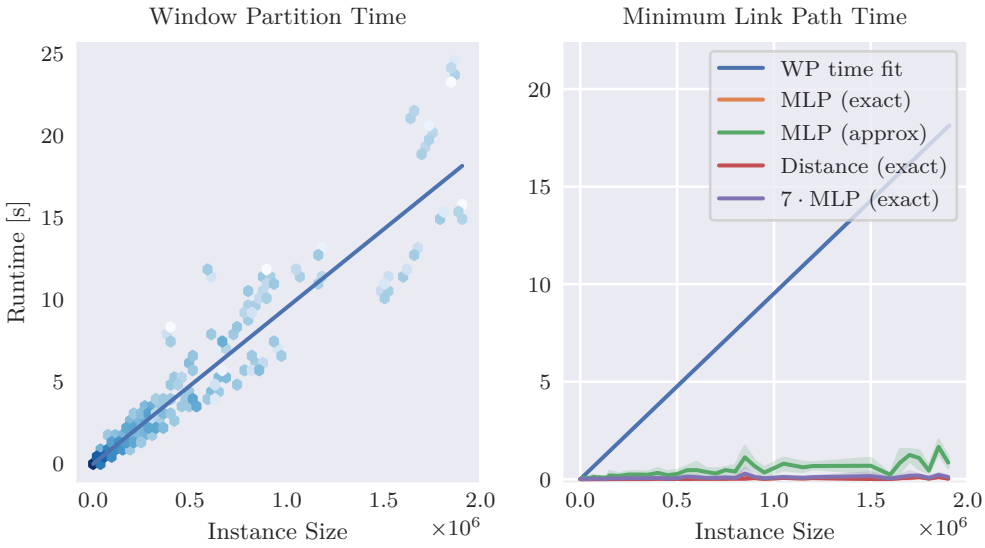


Figure 8 *Left:* The runtime distribution of our window partition implementation, which overall appears to scale linearly, with some instances slightly below and some slightly above the average. *Right:* Comparison of the individual components, showing only the regression line for the window partition time to improve readability. It can be seen that, once a window partition is computed, computing the link distance is very cheap. A single path reconstruction has almost negligible cost, whereas $k = 7$ reconstructions for the approximate approach are more noticable, but still clearly dominated by the window partition time.

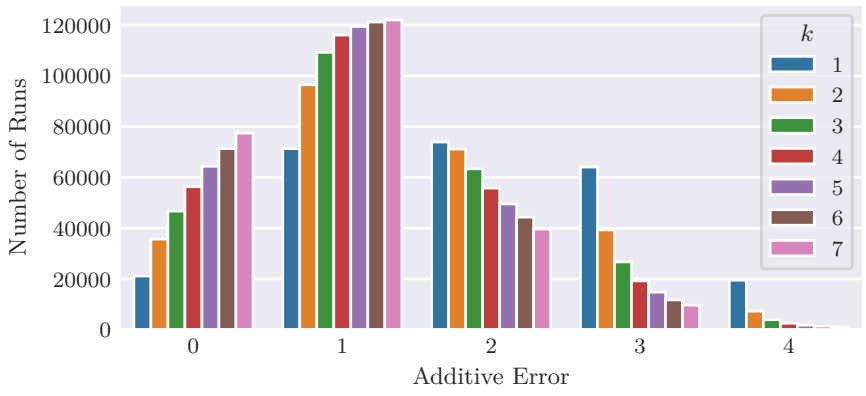


Figure 9 The additive approximation error of our approximate minimum link paths, depending on k, the number of source points used to compute window partitions. In particular, for $k \geq 4$, the majority of queries result in an error of 0 or 1.

An immediate generalization is to non-simple polygons. Suri and O'Rourke [51] studied algorithms for computing visibility polygons with holes. Following the approach described in [10], we can extend our implementation to handle polygons with holes by first splitting the polygon into simple subpolygons and then computing the union of these visibility polygons. The same can be considered for link distance in polygons with holes [42, 40].

Another set of applications arises from using our implementation as a critical tool for dealing with more complex optimization problems that require frequent visibility queries as the critical subroutine. An important example is the Watchman Route Problem (WRP), in which the goal is to find a cheapest tour in a (simple or non-simple) polygon P for a watchman along which every point in P can be seen; here "cheapest" may refer to a minimization of total length or correspond to the number of involved edges. (There are numerous other variants, such as using multiple watchmen.) While a minimum-length watchman tour inside a simple polygon can be computed in polynomial time inside a simple polygon, with the best known time bounds [18] being $\mathcal{O}(n^3 \log n)$ for the "anchored" WRP, in which the tour is required to pass through a specified anchor point, and $\mathcal{O}(n^4 \log n)$ for the "floating" WRP, in which no anchor point is specified; this is only efficient in theory, and we are not aware of any practical implementations. Moreover, the problem is NP-hard for non-simple polygons (with an immediate reduction from the TSP), so practical computation requires more involved exact algorithms that combine the computation of shortest tours with set cover, i.e., many segment visibility computations along the way. This resembles practical approaches for computing optimal solutions for instances of the excruciatingly difficult Art Gallery Problem; as previous work has shown, this may actually hinge on the theoretically "easy" visibility computations instead of the "hard" set cover. For example, of the computation time reported by Kröller, Baumgartner, Fekete and Schmidt [36], 80% was spent on (point) visibility computations: "It is obvious that data structure updates and geometric support procedures are – by far – computationally most expensive, whereas solving the LPs has virtually no runtime impact." See [16] for a deeper dive into how provably optimal solutions for polygons with up to thousands of vertices can be computed. We are optimistic that our newly developed implementation for segment visibility will be a milestone towards similar progress for Watchman Problems, as well as many of the ensuing variants.

References

1 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. *Journal of the ACM*, 69(1):1–70, 2021. doi:10.1145/3486220.

2 Esther M. Arkin, Joseph S. B. Mitchell, and Subhash Suri. Optimal link path queries in a simple polygon. In *Symposium on Discrete Algorithms (SODA)*, pages 269–279, 1992. URL: http://dl.acm.org/citation.cfm?id=139404.139462.

3 Boris Aronov, Leonidas J. Guibas, Marek Teichmann, and Li Zhang. Visibility queries and maintenance in simple polygons. *Discrete and Computational Geometry*, 27(4):461–483, 2002. doi:10.1007/S00454-001-0089-9.

4 Moritz Baum, Thomas Bläsius, Andreas Gemsa, Ignaz Rutter, and Franziska Wegner. Scalable exact visualization of isocontours in road networks via minimum-link paths. *Journal of Computational Geometry*, 9(1):27–73, 2018. doi:10.20382/JOCG.V9I1A2.

5 Jean-Daniel Boissonnat, Olivier Devillers, Monique Teillaud, and Mariette Yvinec. Triangulations in CGAL. In *Symposium on Computational Geometry (SoCG)*, pages 11–18, 2000. doi:10.1145/336154.336165.

6 Prosenjit Bose, Anna Lubiw, and J. Ian Munro. Efficient visibility queries in simple polygons. *Computational Geometry*, 23(3):313–335, 2002. doi:10.1016/S0925-7721(01)00070-0.

7 Reilly Browne, Prahlad Narasimham Kasthurirangan, Joseph S. B. Mitchell, and Valentin Polishchuk. Constant-factor approximation algorithms for convex cover and hidden set in a simple polygon. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1357–1365, 2023. doi:10.1109/FOCS57990.2023.00083.

8 Francisc Bungiu, Michael Hemmer, John Hershberger, Kan Huang, and Alexander Kröller. Efficient computation of visibility polygons. *CoRR*, abs/1403.3905, 2014. arXiv:1403.3905.

9 Mojtaba Nouri Bygi and Mohammad Ghodsi. Weak visibility queries in simple polygons. In *Canadian Conference on Computational Geometry (CCCG)*, 2011. URL: http://www.cccg.ca/proceedings/2011/papers/paper95.pdf.

10 Mojtaba Nouri Bygi and Mohammad Ghodsi. Weak visibility queries of line segments in simple polygons and polygonal domains. *CoRR*, abs/1310.7197, 2013. arXiv:1310.7197.

11 Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6:485–524, 1991. doi:10.1007/BF02574703.

12 Bernard Chazelle. The computational geometry impact task force report: An executive summary. In *Workshop on Applied Computational Geometry (WACG)*, pages 59–65, 1996. doi:10.1007/BFB0014485.

13 Danny Z. Chen and Haitao Wang. Weak visibility queries of line segments in simple polygons. *Computational Geometry*, 48(6):443–452, 2015. doi:10.1016/j.comgeo.2015.02.001.

14 Mark de Berg. On rectilinear link distance. *Computational Geometry*, 1:13–34, 1991. doi:10.1016/0925-7721(91)90010-C.

15 Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008. doi:10.1007/978-3-540-77974-2.

16 Pedro J de Rezende, Cid C de Souza, Stephan Friedrichs, Michael Hemmer, Alexander Kröller, and Davi C Tozoni. Engineering art galleries. In *Algorithm Engineering: Selected Results and Surveys*, pages 379–417. Springer, 2016. doi:10.1007/978-3-319-49487-6_12.

17 Nakju Lett Doh, Chanki Kim, and Wan Kyun Chung. A practical path planner for the robotic vacuum cleaner in rectilinear environments. *IEEE Transactions on Consumer Electronics*, 53(2):519–527, 2007. doi:10.1109/TCE.2007.381724.

18 Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph SB Mitchell. Touring a sequence of polygons. In *Symposium on the Theory of Computing (STOC)*, pages 473–482, 2003.

19 Günther Eder, Martin Held, Steinþór Jasonarson, Philipp Mayer, and Peter Palfrader. Salzburg database of polygonal data: Polygons and their generators. *Data in Brief*, 31:105984, 2020. doi:10.1016/j.dib.2020.105984.

20 Robert Fitch, Zack Butler, and Daniela Rus. 3d rectilinear motion planning with minimum bend paths. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 3, pages 1491–1498. IEEE, 2001. doi:10.1109/IROS.2001.977191.

21 Submir Kumar Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.

22 Andrew Gibiansky. Finger trees, 2014. URL: https://andrew.gibiansky.com/blog/haskell/finger-trees/.

23 Ronald L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972. doi:10.1016/0020-0190(72)90045-2.

24 Leonidas J. Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert Endre Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987. doi:10.1007/BF01840360.

25 Leonidas J. Guibas, John Hershberger, Joseph S. B. Mitchell, and Jack Snoeyink. Approximating polygons and subdivisions with minimum link paths. *International Journal of Computational Geometry and Applications*, 3(4):383–415, 1993. doi:10.1142/S0218195993000257.

26 Leonidas J. Guibas, Edward M. McCreight, Michael F. Plass, and Janet R. Roberts. A new representation for linear lists. In *Symposium on the Theory of Computing (STOC)*, pages 49–60, 1977. doi:10.1145/800105.803395.

27 Mart Hagedoorn and Valentin Polishchuk. Link diameter, radius and 2-point link distance queries in polygonal domains. In *Workshop on Algorithms and Data Structures (WADS)*, pages 34:1–34:15, 2025. ISSN: 1868-8969. doi:10.4230/LIPIcs.WADS.2025.34.

28 Michael Hemmer, Kan Huang, Francisc Bungiu, and Ning Xu. 2D visibility computation. In *CGAL User and Reference Manual*. CGAL Editorial Board, 6.1 edition, 2025. URL: https://doc.cgal.org/6.1/Manual/packages.html#PkgVisibility2.

29 Ralf Hinze and Ross Paterson. Finger trees: a simple general-purpose data structure. *Journal of Functional Programming*, 16(2):197–217, 2006. doi:10.1017/S0956796805005769.

30 Scott Huddleston and Kurt Mehlhorn. A new data structure for representing sorted lists. *Acta informatica*, 17(2):157–184, 1982. doi:10.1007/BF00288968.

31 Hiroshi Imai and Masao Iri. Polygonal approximations of a curve—formulations and algorithms. In *Machine Intelligence and Pattern Recognition*, volume 6, pages 71–86. Elsevier, 1988.

32 Phillip-Raphaël Keldenich, Fabian Kollhoff, Michael Perk, Chek-Manh Loi, and Prahlad Narasimhan Kasthurirangan. Line segment visibility in simple polygons, March 2026. doi:10.5281/zenodo.19194126.

33 Lutz Kettner, Kurt Mehlhorn, Sylvain Pion, Stefan Schirra, and Chee Yap. Classroom examples of robustness problems in geometric computations. *Computational Geometry*, 40(1):61–78, 2008. doi:10.1016/J.COMGEO.2007.06.003.

34 K. Kolarov and B. Roth. On the number of links and placement of telescoping manipulators in an environment with obstacles. In *International Conference on Advanced Robotics (ICAR)*, pages 988–993 vol.2, 1991. doi:10.1109/ICAR.1991.240543.

35 Irina Kostitsyna, Maarten Löffler, Valentin Polishchuk, and Frank Staals. On the complexity of minimum-link path problems. *Journal of Computational Geometry*, 8(2):80–108, March 2017. doi:10.20382/jocg.v8i2a5.

36 Alexander Kröller, Tobias Baumgartner, Sándor P Fekete, and Christiane Schmidt. Exact solutions and bounds for general art gallery problems. *Journal of Experimental Algorithmics*, 17:2–1, 2012. doi:10.1145/2133803.2184449.

37 D. T. Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984. doi:10.1002/NET.3230140304.

38 Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polygons. *Computational Geometry*, 35(1):100–123, 2006. doi:10.1016/j.comgeo.2005.10.005.

39 Anil Maheshwari, Jörg-Rüdiger Sack, and Hristo N. Djidjev. Link distance problems. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 519–558. North Holland / Elsevier, 2000. doi:10.1016/B978-044482537-7/50013-9.

40 Joseph S. B. Mitchell, Valentin Polishchuk, and Mikko Sysikaski. Minimum-link paths revisited. *Computational Geometry*, 47(6):651–667, 2014. Special issue on EuroCG'11. doi:10.1016/J.COMGEO.2013.12.005.

41 Joseph S. B. Mitchell, Valentin Polishchuk, and Mikko Sysikaski. Minimum-link paths revisited. *Computational Geometry*, 47(6):651–667, August 2014. doi:10.1016/j.comgeo.2013.12.005.

42 Joseph S. B. Mitchell, Günter Rote, and Gerhard J. Woeginger. Minimum-link paths among obstacles in the plane. In *Symposium on Computational Geometry (SoCG)*, pages 63–72. ACM, 1990. doi:10.1145/98524.98537.

43 Joseph S. B. Mitchell and Subhash Suri. Separation and approximation of polyhedral objects. *Computational Geometry*, 5:95–114, 1995. doi:10.1016/0925-7721(95)00006-U.

44 Joseph SB Mitchell. Shortest paths and networks. In *Handbook of Discrete and Computational Geometry*, pages 445–466. CRC Publishing, 1997.

45 Joseph SB Mitchell et al. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry*, volume 334, pages 633–702. North Holland, 2000. doi:10.1016/B978-044482537-7/50016-4.

46 Joseph O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, Inc., USA, September 1987.

47 John H. Reif and James A. Storer. Minimizing turns for discrete movement in the interior of a polygon. *IEEE Journal on Robotics and Automation*, 3(3):182–193, 1987. doi:10.1109/JRA.1987.1087092.

48 Jörg-Rüdiger Sack and Jorge Urrutia. *Handbook of computational geometry*. Elsevier, 1999.

49 Subhash Suri. *Minimum link paths in polygons and related problems*. PhD thesis, The Johns Hopkins University, 1987.

50 Subhash Suri. On some link distance problems in a simple polygon. *IEEE Transactions on Robotics and Automation*, 6(1):108–113, 1990. doi:10.1109/70.88124.

51 Subhash Suri and Joseph O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In Alok Aggarwal, editor, *Proceedings of the Second Annual ACM SIGACT/SIGGRAPH Symposium on Computational Geometry, Yorktown Heights, NY, USA, June 2-4, 1986*, pages 14–23. ACM, 1986. doi:10.1145/10515.10517.

52 Ichiro Suzuki and Masafumi Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21(5):863–888, 1992. doi:10.1137/0221051.

53 Mikko Sysikaski. Rectilinear minimum link paths in two and higher dimensions. Master's thesis, University of Helsinki, 2019. Available at https://helda.helsinki.fi/bitstreams/9c8e7b5f-f4ac-42ca-bbc6-bb39a4b24782/download.

54 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987. doi:10.1137/0216030.

55 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 6.1 edition, 2025. URL: https://doc.cgal.org/6.1/Manual/packages.html.

56 Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. CRC press, 2017.

57 David Phillip Wagner. *Path planning algorithms under the link-distance metric*. Phd thesis, Dartmouth College, 2006. Available at https://digitalcommons.dartmouth.edu/cgi/viewcontent.cgi?article=1015&context=dissertations.

58 Hu Xu, Lei Shu, and May Huang. Planning paths with fewer turns on grid maps. In *Proceedings of the International Symposium on Combinatorial Search*, volume 4, pages 193–200, 2013. doi:10.1609/SOCS.V4I1.18298.

A Branch-And-Bound Algorithm for the Traveling Salesman Problem with Difficult Neighborhoods

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
L3S Research Center, Hannover, Germany

Rouven Kniep
Department of Computer Science, TU Braunschweig, Germany

Dominik Krupke
Department of Computer Science, TU Braunschweig, Germany

Michael Perk
Department of Computer Science, TU Braunschweig, Germany

Abstract

The Traveling Salesman Problem with Neighborhoods (TSPN) generalizes the classical Traveling Salesman Problem (TSP) by requiring a tour to visit a set of polygonal regions rather than fixed points, a natural goal that arises in various applications. While the geometric TSP allows arbitrarily close approximation and provably optimal solutions for benchmark instances of significant size, the TSPN is considerably more challenging, both in theory (due to APX-hardness) and practice, for which only benchmark instances up to 16 regions have been solved to optimality.

Here we present a branch-and-bound algorithm that combines a spectrum of geometry-based filters (for reducing the number of considered sequences) with Second-Order Cone Programs (SOCP) (for computing optimal tours for a given permutation of neighborhoods). This allows us to solve larger polygonal TSPN instances than before to within an optimality tolerance of 0.1%; moreover, while previous work (both in theory and practice) relied on relatively benign neighborhoods, we can handle non-convex, non-simple neighborhoods of different sizes. In experiments on 490 benchmark instances with up to 50 polygons each, our method achieves a 99.6 % optimality rate within 300 s, with the remaining two instances solved within 595 s. For 68 larger instances of size $n = 60$, our method still allows solving 86.8 % of instances to optimality within 900 s, leaving only 3 of the instances with optimality gaps above 3 %, with the maximum being 5.53 %.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Discrete optimization; General and reference → Experimentation

Keywords and phrases Geometric optimization, geometric covering, TSP with neighborhoods, exact algorithms, algorithm engineering

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.46

Supplementary Material *Software*: `https://doi.org/10.5281/zenodo.19202809`
Software: `https://github.com/tubs-alg/TSPN-SoCG-2026` [35]
archived at `swh:1:dir:798ae47a3445f53c64575fac631a25a1921a137b`

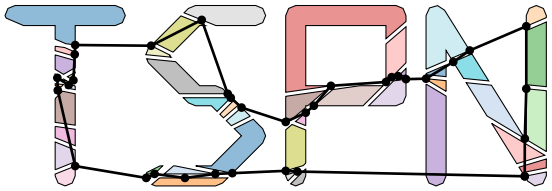
Funding Supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) as part of project *Computational Geometry: Solving Hard Optimization Problems* (CG:SHOP) – 444569951.

Acknowledgements We thank Barak Ugav for his contributions to an earlier version of the branch-and-bound implementation for the CE-TSP, on which parts of the current code are based.

© Sándor P. Fekete, Rouven Kniep, Dominik Krupke, and Michael Perk;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

The Traveling Salesman Problem (TSP) asks for the shortest route that visits a given set of points. Central to many applications, it has also become *the* prototype of a provably difficult, NP-hard optimization problem. A natural geometric generalization of the TSP is the Traveling Salesman Problem with Neighborhoods (TSPN), which asks for a shortest roundtrip that visits each of a given family $P_1, \ldots, P_n$ of regions in the plane; see Figure 1 for an example. This reflects scenarios in which the locations to be visited are not point-sized, but subject to some flexibiliy. Moreover, the TSPN combines the search for optimal touring with aspects of set cover: among others, discrete variants of both the Lawn Mowing Problem (LMP) and the Art Gallery Problem (AGP) can be seen as special cases of the TSPN.



50 polygons, solved in 0.45s

Figure 1 Example TSPN instance with a feasible solution.

Even though both the TSP and the TSPN are NP-hard (making them comparable in theoretical complexity), it can be argued that the TSPN is considerably more challenging. This is clearly reflected in theory: While the geometric TSP allows polynomial-time approximation schemes (Arora [6], Mitchell [38]), APX-hardness of the TSPN was established by de Berg et al. [14]. Moreover, a wide range of sophisticated approximation algorithms rely on assumptions on the shape of the neighborhoods, such as special shapes (e.g., line segments or disks), congruency, convexity, or fatness; see *Related Work* for further details.

A similar gap can be observed on the practical side. The discrete nature of the TSP allows employing a large toolbox of advanced methods from combinatorial optimization and integer programming, enabling provably optimal solutions for point sets of considerable size (such as the 85 900-city instance solved by [3, 12]). On the other hand, the TSPN requires considerable additional effort: even for a given order in which the neighborhoods must be visited, computing a shortest tour is non-trivial. As a consequence, the state of the art until now are provably optimal solutions for benchmark instances up to just 16 regions [29]; see Figure 2 (Left) for an example instance. While hardware and solvers have improved significantly since its publication 13 years ago, solving the problem as a MIQCP with Gurobi on modern hardware did not significantly change the results (see Section 4.6 for details).

Another key difference between discrete TSP in a graph and continuous TSPN in geometry arises from accuracy and error bounds: For a given graph with integer edge weights, verifying that a given bound is met is straightforward; however, in a geometric setting with algebraic complications arising from Euclidean distances and floating-point arithmetic, that is far from clear. The well-known TSPLIB instances [41] use distances rounded to the nearest integers, while even LP solvers such as CPLEX [33] and Gurobi [31] operate with an *optimality tolerance*: The final result is within a given small tolerance of the provable optimum.

Contributions. We present a branch-and-bound algorithm that combines the search for an optimal order for a given set of neighborhoods with Second-Order Cone Programs (SOCP) for computing the best tour for any given sequence of neighborhoods. This significantly

expands the range of instances that can be solved to within an optimality tolerance of 0.1%; note that even the default tolerance of Gurobi is already 0.1‰. Across 490 diverse benchmark instances with up to 50 polygons each, 99.6 % are solved to optimality within a time limit of 300 s, with the remaining two instances solved within 595 s. For 68 larger instances of size $n = 60$, our method still allows solving 86.8 % of instances within 900 s, leaving only 3 of the instances with optimality gaps above 3 %, with the maximum being 5.53 %. Note that $16! \approx 2 \cdot 10^{13}$, while $50! \approx 3 \cdot 10^{64}$ and $60! \approx 8 \cdot 10^{81}$. What is more, we achieve this for non-convex neighborhoods of different sizes, including neighborhoods with holes and even disconnected regions. See Figure 2 for a comparison with the previous state of the art. We also evaluate the impact of various algorithmic design choices, along with geometry-based filters, providing perspectives for even larger instances.

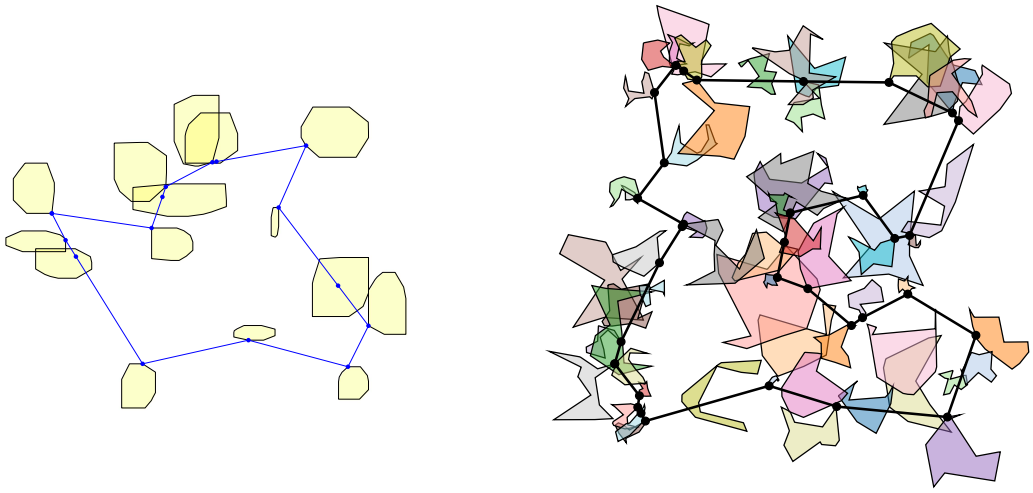


Figure 2 (Left) Previous state of the art [29]: optimal solution for $n = 15$ fat convex regions. (Right) This paper: optimal solution for $n = 60$ non-convex regions of various aspect ratios.

Related Work. A special case of TSPN is the Close-Enough TSP (CE-TSP), where each neighborhood is a circle, which is also related to the Lawn Mowing Problem [4, 5, 26, 27]. Branch-and-bound algorithms have been studied for CE-TSP [19, 13] (possibly with obstacles [15]); however, the TSPN allows arbitrary (including non-convex) neighborhoods, making it more general. A common approach to such problems is to discretize neighborhoods into representative points and solve a Generalized TSP (GTSP) [36], e.g. by a heuristic [32, 42] or an exact method [9, 16]. Advancements in machine learning have also led to unsupervised learning methods for CE-TSP [25, 24]. Many heuristics and approximation algorithms for the TSPN rely on assumptions (e.g., fatness [39, 23], disjoint neighborhoods [14], or comparable region diameters [21]) to manage its APX-hardness. Recently, Antoniadis et al. [2] developed an FPTAS for minimum-perimeter convex polygon intersecting a set of convex objects. Specialized approaches address specific settings such as aerial vehicle routing [34], and hybrid methods combine meta-heuristics with TSP solvers [45]. Non-convex Mixed Integer Nonlinear Programming (MINLP) formulations have been proposed for TSPN, including symmetric and asymmetric variants [29], with algorithmic improvements to reduce solution times. However, computational tests were limited to smaller or convex neighborhoods. Other work derives approximations and bounds for the metric TSPN using the MST with Neighborhoods [11].

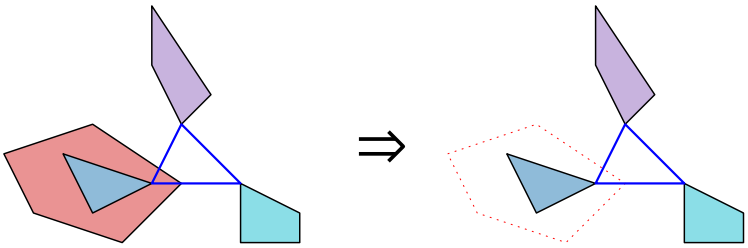


Figure 3 Example of *superset elimination*: a polygon fully contained in another.

Preliminaries. We are given a set of n polygons $\mathcal{P} = \{P_1, \ldots, P_n\}$ in the plane. Polygons may be non-convex and may contain holes. A *tour* is a closed curve in the plane that intersects each polygon $P_i \in \mathcal{P}$ at least once. Without loss of generality, we may assume that an optimal tour is a polygonal cycle visiting one *hitting point* $h(P_i) \in P_i$ for each polygon $P_i \in \mathcal{P}$, connected by straight-line segments. We represent a tour T by the ordered sequence of its hitting points, $(h(P_{\pi(1)}), \ldots, h(P_{\pi(n)}))$, where π is a permutation of $[n]$ and $h(P_{\pi(i)}) \in P_{\pi(i)}$ for all i. The length of T is the sum of the Euclidean distances between consecutive hitting points, including the distance between the last and the first hitting point. Overall, the goal is to find a feasible tour of minimum total length. By triangle inequality, optimal tours cannot self-intersect, so this is a simple polygon of minimum perimeter that intersects all $P_i \in \mathcal{P}$.

2 Preprocessing

Preprocessing is a crucial step in most practical optimization algorithms, as it can significantly reduce the search space and improve algorithmic efficiency. See the literature on MIP presolve [1] and SAT preprocessing [7] for more details on the extensive use of preprocessing in combinatorial optimization. We also investigated and implemented several preprocessing techniques for TSPN instances that either eliminate polygons entirely when any tour that visits the remaining polygons will necessarily cover them, simplify polygons by cutting off unnecessary parts or replacing them with simpler shapes, or add annotations that guide the branch-and-bound algorithm. We first describe basic geometric simplification rules in Section 2.1 and then discuss order annotations in Section 2.2.

2.1 Basic Simplification

We iteratively apply the following three simplification rules until no further reduction is possible. In practice, these rules eliminate nearly all holes and many large polygons.

▶ **Theorem 1** (Superset elimination). *Let $\mathcal{P}$ be a set of polygons in a TSPN instance. If a polygon $P \in \mathcal{P}$ is fully contained in another polygon $P' \in \mathcal{P}$ (see Figure 3), then P' can be safely removed without affecting the optimal tour.*

Proof. Any tour that visits P necessarily intersects P', since P' completely contains P. ◀

▶ **Theorem 2** (Hole removal). *Let $\mathcal{P}$ be a set of polygons for a TSPN instance. Suppose a polygon $P \in \mathcal{P}$ contains a hole h, and there exists a polygon $P' \in \mathcal{P}$ lying entirely outside h (see Figure 4). Then the hole h can be removed from P without affecting the optimal tour.*

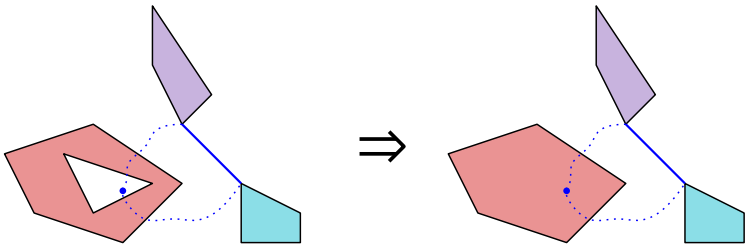


Figure 4 Example of *hole removal*: the hole can be discarded when another polygon lies outside of it. If we were to visit it, we would already intersect the polygon containing it.

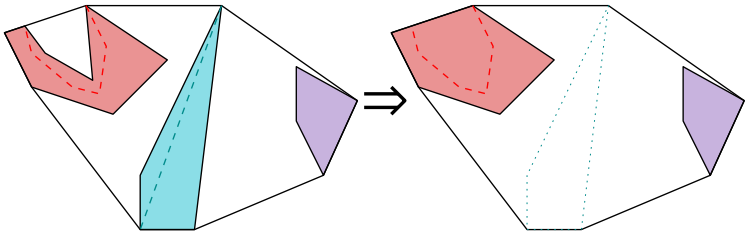


Figure 5 Example of *pocket removal*: of the red polygon, we can remove the pocket, and the blue polygon, we can remove entirely.

Proof. Any tour visiting P' must pass through P, since P' lies outside the hole. If the tour enters the hole h, it already intersects P on the way to or from P'. Thus coverage of h guarantees coverage of P, and the hole does not impose any additional constraint. ◀

▶ **Theorem 3** (Pocket removal). *Let B be a convex closed curve containing an optimal TSPN tour for an instance $\mathcal{P}$, e.g., the convex hull of $\mathcal{P}$. Suppose that a polygon $P \in \mathcal{P}$ contains a path with both endpoints on B, fully lying inside P. This path, together with the arc of B between its endpoints, forms a closed curve that encloses a region (the* pocket*). If at least one polygon of $\mathcal{P}$ lies fully outside the enclosed area, then the enclosed region may be added to P, i.e., the pocket can be removed, without affecting the optimal tour. Furthermore, if also the enclosed region fully contains a polygon from $\mathcal{P}$, then P itself can be safely removed.*

Proof. A tour entering the enclosed region must cross the enclosing path. As the tour must also visit polygons outside the enclosed region, it must cross this path again when leaving. Each crossing intersects P, so P is inevitably covered. If the enclosed region also contains polygons from $\mathcal{P}$, the tour must enter and leave the region, crossing the path at least twice. Thus, removing the pocket preserves feasibility, and P becomes redundant. ◀

Note that the second case of Theorem 3 is automatically handled by Theorem 1 once the pocket has been removed from the polygon, because the enclosed polygon will be fully contained in the modified polygon. If B is not the convex hull, we may first intersect all polygons with B. This ensures that pockets adjacent to B are also removed consistently and that the simplification rules behave exactly as in the convex-hull case.

2.2 Order Annotations

Our algorithm suffers from a large branching factor, as it considers every possible ordering of input polygons. Fortunately, geometric properties of optimal tours let us prune many insertion choices, in particular for polygons that intersect the boundary of a tour-enclosing

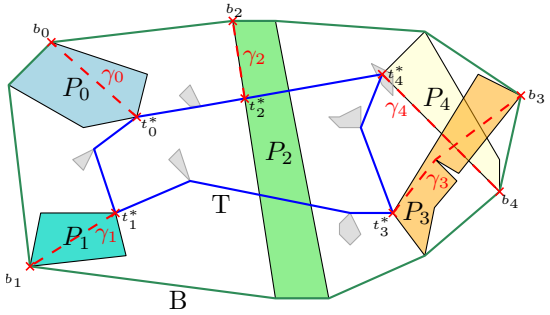


Figure 6 Illustration for b and t^* on B and T. Note that P_3 and P_4 overlap, so we can have $\mathcal{P}'_B = \{P_0, P_1, P_2, P_3\}$ or $\mathcal{P}'_B = \{P_0, P_1, P_2, P_4\}$ as maximal sets. While preprocessing would remove P_2 (via pocket removal and superset elimination, cf. Figure 5), we retain it here to emphasize that Theorem 5 holds independently of preprocessing.

hull B, such as the convex hull of all polygons $\mathcal{P}$. A key property is that an optimal tour must respect the order of the polygons along this hull. Although this is intuitive, complications arise when polygons overlap, because the order is then only partially defined. Let $\sigma_B : [|\mathcal{I}_B|] \to \mathcal{I}_B$ be a cyclic order on a subset $\mathcal{I}_B \subseteq [n]$ of indices. The relation $\circlearrowleft_{\sigma_B}(i,j,k)$ is satisfied if either index is not in $\mathcal{I}_B$ or polygons P_i, P_j, P_k are not pairwise disjoint, or if in the cyclic order of σ_B, index j lies between indices i and k following the fixed orientation of σ_B.

▶ **Definition 4.** *Let $\sigma_B : [|\mathcal{I}_B|] \to \mathcal{I}_B$ be a cyclic order on a subset $\mathcal{I}_B \subseteq [n]$ of indices, then $\circlearrowleft_{\sigma_B}(i,j,k)$ is defined as*

$$\circlearrowleft_{\sigma_B}(i,j,k) = \begin{cases} 1 & \text{if } \{i,j,k\} \nsubseteq \mathcal{I}_B \vee P_i \cap P_j \neq \emptyset \vee P_j \cap P_k \neq \emptyset \vee P_i \cap P_k \neq \emptyset \\ 1 & \text{if } (\sigma_B^{-1}(j) - \sigma_B^{-1}(i)) \bmod |\mathcal{I}_B| \leq (\sigma_B^{-1}(k) - \sigma_B^{-1}(i)) \bmod |\mathcal{I}_B| \\ 0 & \text{otherwise} \end{cases}.$$

▶ **Theorem 5.** *Let B be a simple closed convex curve that contains an optimal TSPN tour for a given instance. Let $\mathcal{I}_B \subseteq [n]$ and $\mathcal{P}_B = \{P_j : j \in \mathcal{I}_B\}$ be the set of polygons intersecting B, and for each polygon fix the reference points $\{b_j \in P_j \cap B \mid j \in \mathcal{I}_B\}$ which define a cyclic order σ_B on $\mathcal{I}_B$ that corresponds to the order of the reference points along B. Then there exists an optimal tour T with visiting order π such that, for every $a < b < c \in [n]$ the order of their hitting points coincides with the order along B, i.e. $\forall a < b < c \in [n] : \circlearrowleft_{\sigma_B}(\pi(a), \pi(b), \pi(c))$.*

Proof. Let T be an optimal tour contained in a simple closed convex curve B. Both T and B are counterclockwise oriented. For each $P_j \in \mathcal{P}_B$ we fix a reference point $b_j \in P_j \cap B$ which is used to define the order σ_B on $\mathcal{I}_B$. As T is feasible it must visit all polygons; for all $j \in \mathcal{I}_B$ we choose the hitting point $t_j^* \in P_j \cap T$ minimizing the geodesic distance in P_j to b_j. Let γ_j be a shortest path in P_j from b_j to t_j^* and denote γ_j^{-1} its reversal. By convexity of B, γ_j meets B only at b_j; by minimality, γ_j meets T only at t_j^*; and since $\gamma_j \subseteq P_j$, it can intersect $\gamma_{j'}$ only if $P_j \cap P_{j'} \neq \emptyset$, see Figure 6.

Let $\phi : [m] \to \mathcal{I}_B$ be an injection with $m \leq n$ that defines an ordered subsequence (induced by σ_B) corresponding to pairwise disjoint polygons, i.e., $\forall i \neq j \in [m] : P_{\phi(i)} \cap P_{\phi(j)} = \emptyset$ and $\forall i < j < k \in [m] : \circlearrowleft_{\sigma_B}(\phi(i), \phi(j), \phi(k))$. We write $[\cdot,\cdot]_B$ for the arc of B, and $[\cdot,\cdot]_T$ for the subpath of T between two points following B's and T's orientation respectively. For each $i \in [m]$ we define the closed curve Γ_i between $b_{\phi(i)}$ and $b_{\phi(i+1)}$ (indices should be seen modulo m) as $\Gamma_i = [b_{\phi(i)}, b_{\phi(i+1)}]_B \cup \gamma_{\phi(i+1)} \cup [t^*_{\phi(i+1)}, t^*_{\phi(i)}]_T^{-1} \cup \gamma_{\phi(i)}^{-1}$, which bounds a simply connected region R_i. Thus, each region boundary is explicitly composed of: (i) the

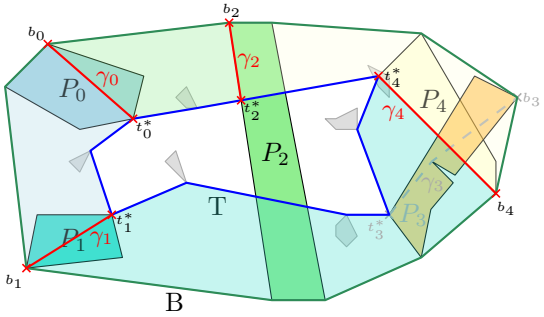


Figure 7 The paths of $\mathcal{P}'_B = \{P_0, P_1, P_2, P_4\}$ tile the area between B and T, forming a cycle of regions and boundaries. We can also see that it is important to select a pairwise disjoint set of polygons, as the crossing paths of P_3 and P_4 would break the argument otherwise. While one can find an alternative b_3 to avoid this, we need to remember that we actually do not know T in advance but provide an argument that works for any T.

arc of B, (ii) the two shortest paths $\gamma_{\phi(i)}, \gamma_{\phi(i+1)}$, and (iii) the corresponding subpath of T. These regions are interior-disjoint and tile exactly the annular portion between B and T, see Figure 7. Each shortest path $\gamma_{\phi(i)}$ with $i \in [m]$ has two incident regions, one on each side which means that the regions form a cycle around T.

Tracing T along the subpaths $[t^*_{\phi(i)}, t^*_{\phi(i+1)}]_T$ implies that T is a concatenation of hitting points that respect $\circlearrowleft_{\sigma_B}(\cdot,\cdot,\cdot)$ for any valid injection ϕ which implies that $\circlearrowleft_{\sigma_B}(\cdot,\cdot,\cdot)$ holds for all triples of pairwise disjoint polygons in $\mathcal{P}_B$. Degenerate contacts between T and B (or degenerate γ_j) can be removed by an infinitesimal perturbation preserving optimality and incidences. Hence, for any $\circlearrowleft_{\sigma_B}$, there exists an optimal tour T for which every triple $P_{\pi(a)}, P_{\pi(b)}, P_{\pi(c)} \in \mathcal{P}_B$ with $a < b < c$ must satisfy $\circlearrowleft_{\sigma_B}(\pi(a), \pi(b), \pi(c))$. ◀

During preprocessing, we annotate each polygon $P_i \in \mathcal{P}_B$ intersecting the convex hull B with its position in the cyclic order σ_B, together with the indices of all polygons $P_j \in \mathcal{P}_B$ that overlap with P_i. This information enables us to select a maximal subset of pairwise disjoint polygons $\mathcal{P}'_B \subseteq \mathcal{P}_B$, which induces a total order that any optimal tour can respect.

3 Branch-and-Bound Algorithm

Our branch-and-bound algorithm (Algorithm 1) explores the space of polygon orderings by incrementally building sequences: starting from a small initial subsequence that is guaranteed to be extendable to an optimal solution (Section 3.2), it repeatedly selects a polygon not yet in the sequence and creates a child node for each possible insertion position, forming a search tree (Section 3.3). For each candidate sequence, a Second-Order Cone Program (SOCP) computes the optimal tour for that ordering (Section 3.1). If this tour visits all polygons, it is a feasible solution (see Figure 8); otherwise, the SOCP provides a lower bound on any tour extending this partial sequence – if it exceeds the best known solution, no completion can improve upon it and the entire subtree is pruned. This builds on the framework of Coutinho et al. [13] for the CE-TSP, which we improve and extend to handle the TSPN.

Throughout the search, the algorithm keeps track of the incumbent (best known feasible) solution and the node with the lowest lower bound, which together define the current optimality gap. As the search tree can grow large, the order in which nodes are evaluated can have significant impact on performance: finding good feasible solutions early strengthens

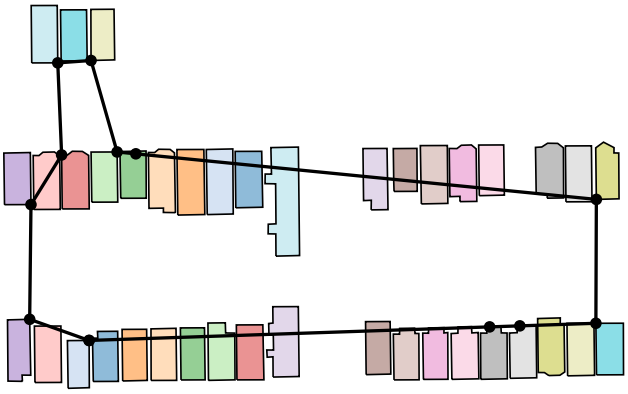


Figure 8 Illustration of implicit coverage in the TSPN. Many polygons will be covered naturally when connecting other polygons, without needing to explicitly model them in the optimization.

pruning, while evaluating nodes with low lower bounds tightens the optimality gap. A search strategy balances these objectives (Section 3.4). Search is terminated if this gap is smaller than a desired threshold, or if all nodes are explored.

3.1 Touring a Sequence of Polygons

By Dror et al. [20], the Touring Polygons Problem (TPP) that asks for the shortest tour visiting a given sequence of polygons can be solved in polynomial time for convex polygons when the visitation order and a start and end point are fixed. The runtime of the algorithm was later improved by Tan and Jiang [43] to $O(k^2m)$ for k convex regions with a total number of m vertices. If the start and end points are not fixed, we can still solve it in polynomial time using a Second-Order Cone Program (SOCP), similar to Coutinho et al. [13].

▶ **Theorem 6.** *Let $P_0, \dots, P_{n-1} \subset \mathbb{R}^2$ be convex polygons. Then the shortest tour visiting these polygons in order can be computed in polynomial time.*

Proof. A convex polygon P_i can be represented by linear constraints $S_i(x, y)$. We formulate the problem as a Second-Order Cone Program (SOCP), which can be solved in polynomial time via interior-point methods [8]. For each i, introduce a point $(x_i, y_i) \in \mathbb{R}^2$ constrained by $S_i(x_i, y_i)$, ensuring $(x_i, y_i) \in P_i$. For readability, all indices are assumed modulo n.

To encode the total tour length, let $d_i \geq 0$ be the distance between (x_i, y_i) and (x_{i+1}, y_{i+1}). We introduce auxiliary variables $\hat{x}_i, \hat{y}_i$ subject to $\hat{x}_i \geq x_i - x_{i+1}$, $\hat{x}_i \geq x_{i+1} - x_i$, $\hat{y}_i \geq y_i - y_{i+1}$, $\hat{y}_i \geq y_{i+1} - y_i$, and impose second-order cone constraints $d_i^2 \geq \hat{x}_i^2 + \hat{y}_i^2$.

All constraints are linear or second-order cone constraints, so the formulation is an SOCP of polynomial size. Therefore, the shortest tour can be computed in polynomial time. ◀

As we ultimately aim to solve TSPN instances with non-convex polygons (and potentially holes if preprocessing does not eliminate them), we must extend the SOCP approach. Dror et al. [20] showed that the problem becomes NP-hard for non-convex polygons, so no polynomial-time algorithm can be expected in general. We propose two strategies, both based on convex decompositions, which partition a non-convex polygon into convex components, see Figure 9.

The first approach is to replace the SOCP by a Mixed-Integer SOCP (MISOCP). For a non-convex polygon P_i', let $\mathcal{R}_i$ be its convex decomposition. Introduce a binary variable $r_c \in \mathbb{B}$ for each convex piece $c \in \mathcal{R}_i$ and enforce $\sum_{c \in \mathcal{R}_i} r_c = 1$. Let $S_i^c(x, y)$ be the linear constraints

Algorithm 1 Branch-and-Bound Algorithm.

Require: Set of polygons $\mathcal{P}$
Preprocess and simplify $\mathcal{P}$.
$\mathcal{Q} \leftarrow [\,]$ ▷ Queue of leaf nodes to explore
$v \leftarrow \textsc{getRootNode}(\mathcal{P})$ ▷ See Section 3.2
Compute optimal tour and lower bound on v via SOCP ▷ See Section 3.1
$\mathcal{Q}$.push(v)
$ub \leftarrow \infty$
while $\mathcal{Q} \neq \emptyset \wedge \min\{v'.lb \mid v' \in \mathcal{Q}\} < ub$ **do**
    $v \leftarrow \textsc{getNextNode}(\mathcal{Q})$ ▷ Search strategy, see Section 3.4
    Remove v from $\mathcal{Q}$
    if $v.lb \geq ub$ **then**
        continue
    end if
    if Solution in v covers all polygons in $\mathcal{P}$ **then**
        $ub \leftarrow v.lb$
    else
        for *child* $\in \textsc{Branch}(v)$ **do** ▷ Branching, see Section 3.3
            Compute optimal tour and lower bound on *child* via SOCP
            $\mathcal{Q}$.push(*child*)
        end for
    end if
end while
return Incumbent solution corresponding to ub, or $\perp$ if no solution was found.

defining ∂c. To ensure $(x, y) \in P'_i$, impose the implications $r_c = 1 \Rightarrow S^c_i(x, y) \quad \forall c \in \mathcal{R}_i$. These implications can be modeled using indicator constraints (when supported) or with Big-M formulations, where M can be bounded using the polygon's bounding box. Both indicator and Big-M constraints tend to produce weak relaxations for the MISOCP solver, which provides limited guidance to its internal branch-and-bound. To improve the relaxation, one may additionally include the convex-hull constraints $S_i(x, y)$ of P'_i, which are redundant but can strengthen the continuous relaxation.

The second approach retains the efficient SOCP formulation and incorporates convex decomposition only through branching. Instead of encoding the convex components by binary variables, the branch-and-bound tree explicitly chooses which convex part of a non-convex polygon the tour must hit. This extends the branching decision from the visitation order

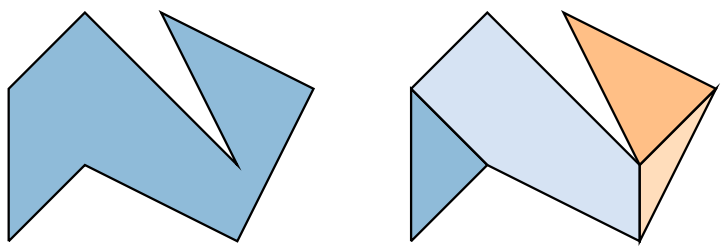


Figure 9 A non-convex polygon (left) and its decomposition in convex areas (right).

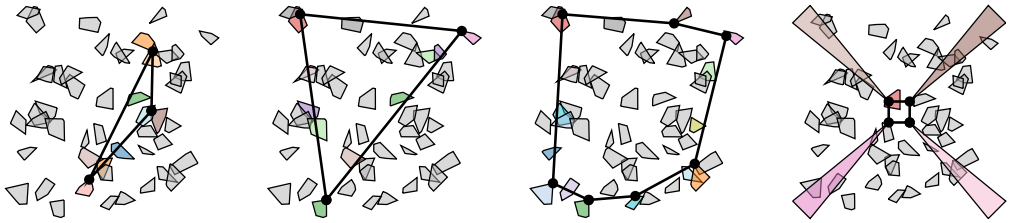


Figure 10 Root node strategies (left to right): Random, LEFP, CHR, and an example demonstrating poor CHR performance.

to the choice of the convex component, while avoiding indicator and Big-M constraints entirely. As a result, this approach avoids the weak relaxations that appear in the MISOCP formulation, although it may increase the size of the search tree.

For polygons without holes, a minimum convex decomposition can be computed in polynomial time [30, 10]. For polygons with holes, a decomposition is always possible, for example by triangulation, but finding a minimal decomposition is NP-hard [37]. In practice, our preprocessing eliminates all holes from the benchmark instances, so the NP-hard minimal decomposition is not needed. The theoretical caveat remains for degenerate instances where hole removal cannot apply, e.g., when every other polygon lies inside the hole.

Finally, any non-convex polygon can either weaken the relaxation in the MISOCP approach or introduce substantial branching overhead. However, in optimal tours we often observe that visitation points of spanning polygons, meaning those not implicitly covered by connections between other polygons, tend to lie on extreme points, that is, on the convex hull of the polygon. This motivates a lazy strategy in which every polygon is initially replaced by its convex hull, making all polygons appear convex in the relaxation. Only when a computed visitation point lies outside the true polygon do we enforce its full non-convex geometry.

3.2 Root Node

The root node's initial polygon sequence must be extendable to an optimal solution. Any sequence of up to three polygons trivially satisfies this condition. We evaluate four strategies: **Random** selects three polygons at random. **Longest Edge (LE)** selects the two polygons with largest pairwise distance. **Longest Edge+Farthest Polygon (LEFP)** extends LE by adding the polygon farthest from both. **Convex Hull (CHR)** selects polygons based on their convex hull positions.

Using the observations from Section 2.2, we can construct larger initial sequences: for any set of pairwise disjoint polygons intersecting the convex hull, σ_B defines a total order that an optimal solution can respect. Finding a cardinality-maximal set of such polygons is an instance of the NP-hard *maximum independent set* problem, but optimality is not required here; a greedy heuristic yields a sufficiently large set in practice. This preserves completeness while often producing significantly stronger initial relaxations. Although it is possible to construct artificial instances where this strategy produces arbitrarily weak relaxations, for example by placing polygons on the convex hull that protrude far inward and thereby induce an initial tour of negligible length, such instances appear to be rare in practice. Figure 10 illustrates three of the strategies, including an example where CHR performs poorly.

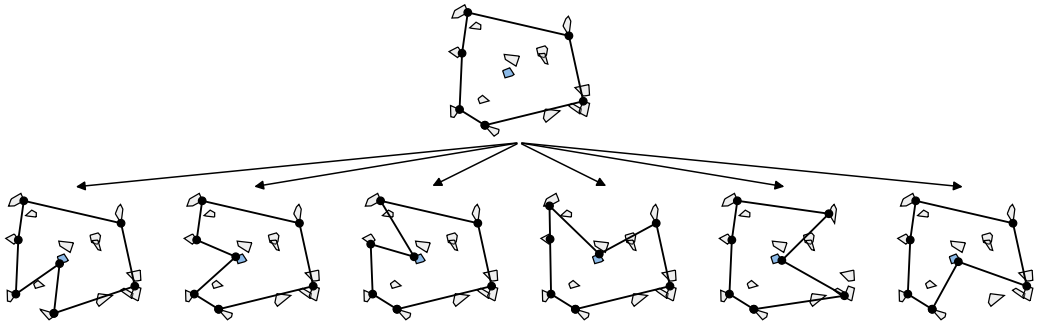


Figure 11 Branching on the insertion position of the selected farthest polygon.

3.3 Branching

When the (partial) tour in a node does not cover all polygons, we branch by selecting a missing polygon and creating a new branch for each possible insertion position. In our implementation, we select the polygon that lies farthest from the current tour, see Figure 11.

We consider three modeling strategies for handling non-convex polygons. **Eager indicator-based modeling (EIM)** models non-convex polygons explicitly using indicator variables in a MISOCP formulation. Because convex hulls often contain the extremal points that serve as optimal hitting points, we propose a lazy strategy that initially replaces each polygon by its convex hull and only models it as non-convex when necessary. Both of the following strategies employ this lazy approach. **Indicator-based modeling (IM)** uses the same indicator-variable formulation as EIM but only introduces it when the convex hull replacement is insufficient. **Decomposition branching (DB)** avoids indicator variables entirely by decomposing the polygon into convex parts and creating one branch per part, see Figure 12.

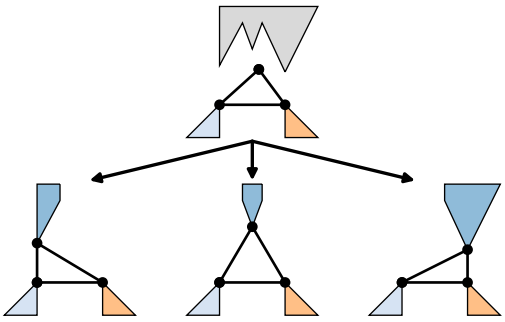


Figure 12 Branching on a non-convex polygon by decomposing it into convex pieces.

3.4 Search Strategies

We implement four strategies to select the next node for exploration during the search.

Best First Search (BFS) selects the node with the smallest lower bound, improving the global bound quickly. However, incomplete sequences tend to have shorter estimated lengths, so BFS often prioritizes them over full coverage. **Depth First Search (DFS)** continues

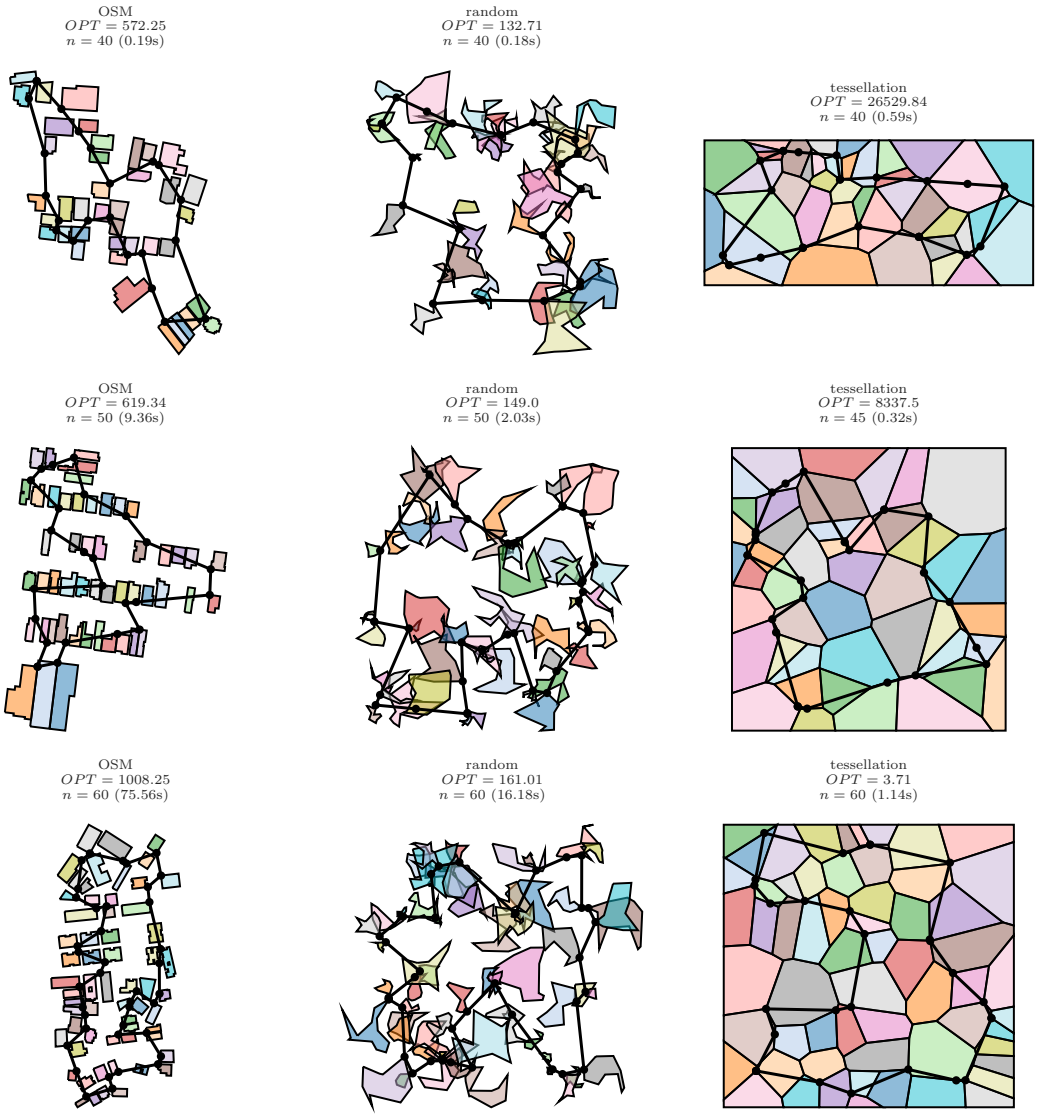


Figure 13 Example solutions within 0.1 % of optimality for instances of size $n \in \{40, 50, 60\}$.

exploring the most promising child, quickly reaching feasible solutions. Good incumbents strengthen pruning, though DFS largely ignores lower bounds. **DFS+BFS** behaves like DFS, but reorders the queue by lower bounds whenever a node is pruned or a new solution is found, combining rapid incumbent improvements with lower-bound tightening.

4 Empirical evaluation

Our algorithm is implemented in C++ and uses Gurobi 12.0 [31] to solve the SOCP and MISOCP formulations. Geometry operations rely on Boost.Geometry 1.83 [28] and CGAL 6.0.1 [44] with exact predicates and constructions. We compiled using g++ 13.3 and ran all experiments on an AMD Ryzen 7900 workstation with 96 GiB of DDR5-5200 RAM under Ubuntu 24.04. We use the term *optimality gap* to refer to the relative difference between the best known upper bound and the global lower bound, i.e., $^{ub}/_{lb} - 1$. The optimality tolerance is the maximum allowed optimality gap after which the algorithm can terminate.

4.1 Research questions

Our experimental evaluation aims to answer the following questions.

Q1 How does the choice of root node sequence affect runtime across instance types?

Q2 Which search strategy (DFS, BFS, or a combined DFS+BFS) offers the best trade-off between time-to-solution and bound improvement?

Q3 For non-convex polygons, does decomposition branching outperform indicator-based modeling in practice with respect to runtime and relaxation quality?

Q4 How does relaxing the optimality tolerance affect the performance and how does our approach scale with increasing instance sizes?

4.2 Experiment design

To answer our questions, we collected and generated a large set of 558 instances, consisting of instances from the following instance classes. In all cases, the coordinates of points in the instances are either integers or double precision floating-point numbers.

random We generated 20 random instances with $n \in \{5, 10, 15, 20, 30, 40, 50, 60\}$ random polygons with roughly 45 % concave points and between 10 and 15 edges. Polygons may overlap and may have holes. This resulted in 160 random instances.

tessellation Instances derived from well-known publicly available benchmark instance sets. To obtain polygonal regions, we applied a Voronoi tessellation to point sets from various sources. These include point sets previously used in the CG:SHOP challenges [18, 17], TSPLIB instances [41], and point sets from the Salzburg Database of Polygonal Inputs [22]. In total we generated 78 instances with $n \leq 60$ polygons.

OSM Instances derived from *Open Street Map* (OSM) data [40]. Our dataset covers building footprints from 20 major cities. We used a quadtree-like subdivision to achieve desired sizes of $n \in \{5, 10, 15, 20, 30, 40, 50, 60\}$ and then randomly sampled two instances from each city in our collection. This resulted in a total of 320 instances.

All instances were preprocessed with the techniques from Section 2, resulting in 360 (out of 558) altered instances with no holes. This took 0.01 s on average, with a maximum of 0.06 s. The resulting instances show reduced complexity. In particular, the number of convex pieces (i.e., the number of convex components in an optimal convex decomposition) decreased by an average of 2.9 per instance. Example instances and their optimal solutions can be found in Figure 13. Unless stated otherwise, all experiments use the full set of 558 instances.

4.3 RQ1: Root node selection

The choice of the root node sequence can significantly influence the search performance. A larger root sequence yields a tighter initial lower bound and fixes the relative order of more polygons. However, it may include polygons that would otherwise be implicitly covered by the tour (see Figure 8), unnecessarily enlarging the explicit sequence and increasing both the branching factor and the SOCP size at every node. We compared four strategies for selecting the root node sequence with a time limit of 60 s and 1 % optimality tolerance. CHR selects polygons based on the convex hull order, LE selects two polygons that are farthest apart, LEFP extends LE by adding a third polygon that maximizes the distance to the first two, and Random selects a random sequence, see Section 3.2 for details. Figure 14 shows the number of instances and the optimality gaps after 60 s. Overall, using a CHR and LEFP strategy appear to be best across our benchmarks with solutions within 1 % tolerance found

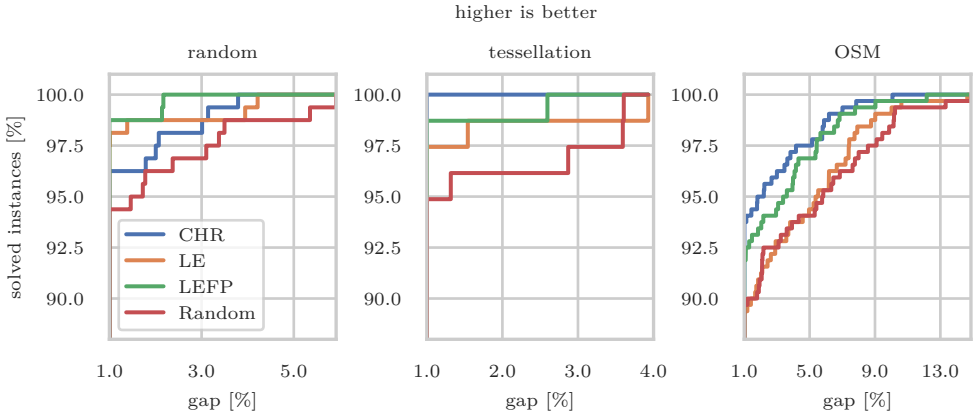


Figure 14 Gaps for different root node strategies after 60 s with 1 % optimality tolerance, separated by instance type. CHR and LEFP outperform the others, with CHR strongest on OSM and tessellation instances and LEFP more robust on random instances.

in 95.16 % and 94.80 % of the instances, respectively. LEFP was more robust for instances that contain many intersections (such as our random instances), while CHR performed best on average for the OSM and tessellation instances which have many polygons on their convex hull. LE performed reasonably well but suffers from bad initial lower bounds. Random root selection is not recommended due to high variance and inferior median performance.

4.4 RQ2: Search strategy

The choice of search strategy affects how quickly solutions are found and how fast the global lower bound improves. We compared the three strategies from Section 3.4 – DFS, BFS, and combined DFS+BFS – with a time limit of 60 s for optimality tolerances of 1 % and 5 %. The outcomes are shown in Figure 15. We observe that for 5 % optimality tolerance, DFS and DFS+BFS have similar performance with about 98 % of the instances solved to optimality, while DFS+BFS is the winner for smaller tolerances with about 95 % of the instances solved to 1 % optimality. DFS achieves the fastest time-to-first-solution in most instances, while BFS improves the global lower bound more quickly. The combined DFS+BFS provides a good trade-off, particularly when slightly relaxed optimality tolerances are acceptable (it often terminates quickly for moderate gaps). For the remainder of our experiments, we use the DFS+BFS strategy as it provides the best overall performance.

4.5 RQ3: Decomposition branching

We evaluate the performance of the three modeling strategies for non-convex polygons introduced in Section 3.3. IM represents non-convex regions using indicator variables, which can result in long solve times at individual nodes. In contrast, DB avoids indicator variables but increases the size of the search tree, potentially leading to more branching. In both DB and IM, each polygon is initially replaced by its convex hull and only treated as non-convex when necessary. EIM does not employ this lazy strategy and models all non-convex polygons explicitly from the outset. We compared all approaches with a time limit of 60 s and an optimality tolerance of 1 %, excluding the purely convex instances (480 instances remaining). The results are shown in Figure 16. DB and IM exhibit very similar performance. While DB is slightly faster on some instances, it solves fewer instances to within 1 % optimality within

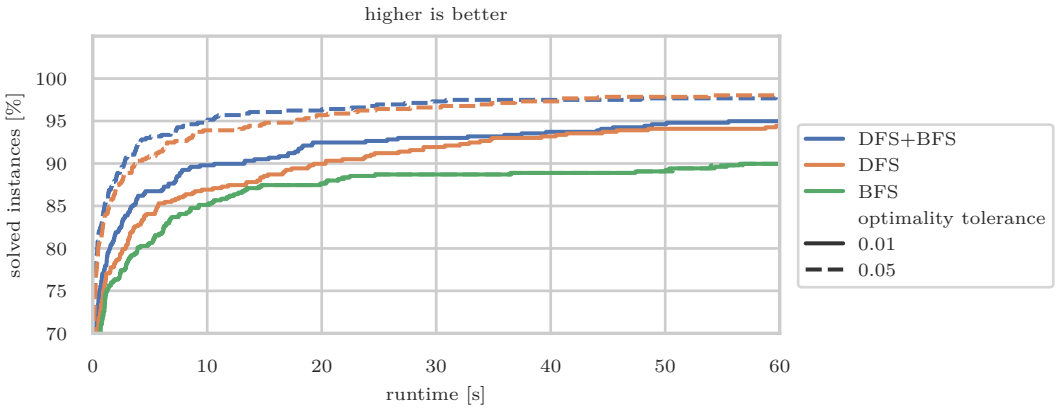


Figure 15 Instances solved within 60 s for different search strategies with 1 % and 5 % optimality tolerance. DFS+BFS offers the best trade-off between fast solutions (DFS) and rapid lower-bound improvement (BFS).

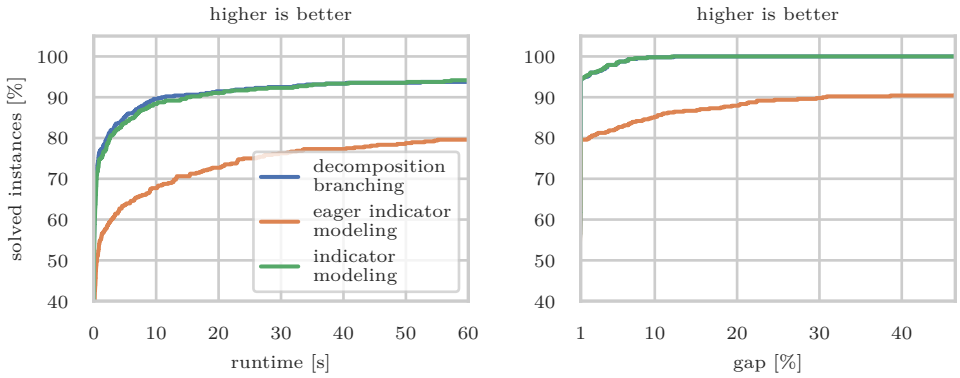


Figure 16 (Left) Runtime for solving instances with decomposition branching as well as eager and non-eager indicator-based modeling within a time limit of 60 s and 1 % optimality tolerance. (Right) Optimality gaps for the remaining instances. Decomposition branching has a slightly better performance than indicator-based modeling, and eager indicator-based modeling performs worse.

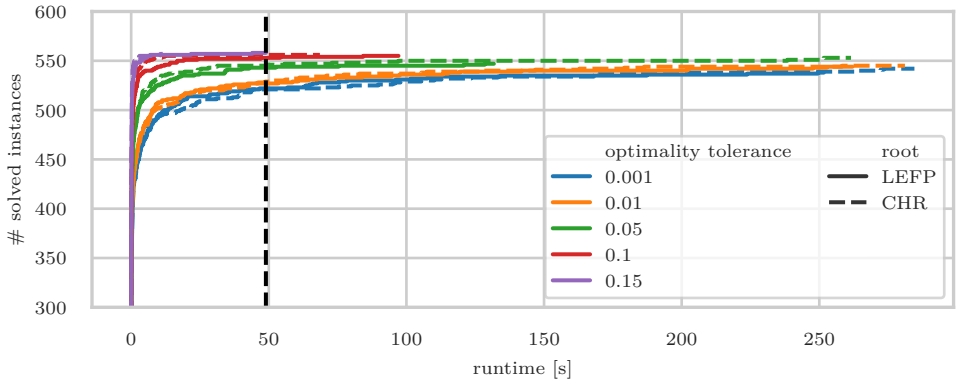


Figure 17 Number of instances solved within a given optimality tolerance within a given time. Wider gaps of 5 % to 15 % substantially reduce computation time, while for smaller gaps of 0.1 %, not all instances could be solved within the time limit.

the time limit (93.75 % for DB vs. 94.17 % for IM). In contrast, EIM performs substantially worse, solving only 79.58 % of instances within the time limit. Overall, these results support the use of the lazy initialization strategy for non-convex polygons. Because DB and IM perform similarly but DB avoids the need for indicator variables, we adopt decomposition branching as our default approach for the remainder of the experiments.

4.6 RQ4: Scalability with optimality tolerance

Relaxing the optimality tolerance is often acceptable in practical applications and can significantly reduce computation time in exact optimization. Gentilini et al. [29] reported optimal solutions for up to 16 convex regions which impose less complexity than our general polygonal regions. We evaluated the performance of both the CHR and LEFP root node strategies with DFS+BFS search for optimality tolerances of 0.1 %, 1 %, 5 %, 10 %, and 15 % with a time limit of 300 s. This instance set comprises 558 instances with sizes ranging from $n = 5$ to $n = 60$ polygons of varying complexity. The number of solved instances after a given time is shown in Figure 17. We observe negligible differences between 0.1 % and 1 %, whereas tolerating 5 %, 10 % or 15 % often yields large savings in solve time and increases the number of instances solved. For 0.1 % tolerance, runs using the CHR root node strategy yielded the best results with 97.13 % of all instances solved to optimality in less than 300 s. The instances not solved to optimality had either $n = 50$ or $n = 60$ polygons. More precisely, for $n = 50$, only 2 out of 65 instances were not solved to optimality within the time limit, with a maximum optimality gap of 0.83 %. Both instances can however be solved to the desired tolerance within 595 s and 368 s, respectively. The remaining 14 unsolved instances of size 60 all had gaps within 7 %. A subsequent run with a time limit of 900 s solved 5 additional instances to 0.1 % optimality, leaving 9 instances unsolved with a maximum gap of 5.53 %. This amounts to a total of 98.39 % of all instances solved to optimality within 900 s. Relaxing the requirements for an optimal solution to 15 % allows the LEFP strategy to solve all instances within less than 50 s.

Comparison to previous work. Prior to our geometric branch-and-bound approach, TSPN was commonly modeled as a MINLP and solved using general-purpose MINLP solvers. In these formulations, subtour elimination constraints are typically enforced through a cutting-plane framework where a relaxation of the problem is solved, subtours are detected for example via max-flow separation, and violated constraints are added as they are identified.

A representative example of this line of work is the method of Gentilini et al. [29]. Their approach integrates COUENNE, Ipopt, and CPLEX, and incorporates TSPN-specific strengthening techniques within COUENNE. Nevertheless, the method remains fundamentally based on the classical MINLP formulation and therefore inherits its high-dimensional spatial branching behavior. Empirical results reported in their study indicate practical tractability primarily for relatively small instances, typically with $n \leq 30$ within a few percent optimality gap, but on older hardware and with less efficient solvers than those available today.[1]

Our approach differs both conceptually and algorithmically. Instead of branching in the high-dimensional MINLP formulation, we perform branch-and-bound directly in the underlying planar geometry. Node relaxations are solved as SOCPs, allowing the algorithm to exploit the polynomial-time solvability of convex subproblems. This design shifts the computational effort from generic nonlinear spatial branching toward geometry-aware decisions.

4.7 Threats to Validity

All solutions were validated, and a set of unit tests ensured correctness of core components. Results were checked for consistency between objective values and lower bounds across different runs. Feasibility of the resulting tours was independently verified after the optimization. However, as with any empirical evaluation, there are threats to validity. Implementation bugs may affect results, though we took care to minimize this risk through testing and validation. The choice of Gurobi as the underlying SOCP solver may influence performance; different solvers could yield varying results. Due to the nature of these solvers, all computations were performed using floating-point arithmetic, which may introduce numerical inaccuracies. Finally, the selected instance sets, while diverse, may not cover all possible scenarios encountered in practice.

5 Conclusion and Future Work

We presented significant practical progress for the TSPN with arbitrary neighborhoods of differing sizes, not only picking the best out of $n!$ permutations, but also computing the best tour for a given order of regions; even for k convex regions with m vertices and given start and end point, the fastest known algorithm for this task requires a time of $O(k^2m)$ [43].

At this point, we are able to solve all instances in our benchmark set of size up to $n = 50$ to within 0.1 % of optimality within reasonable time, and 86.8 % of the larger instances of size $n = 60$, with small optimality gap for the remaining ones. With our current techniques, this size appears to be the threshold between manageable and intractable instances without resorting to runtimes in the order of hours.

This raises the question of how to extend this range by employing more advanced geometric ideas. We are optimistic that further progress is possible, e.g., by using enhanced pruning strategies and additional cuts during the branch-and-bound, making use of further theoretical insights into the structure of locally optimal trajectories.

[1] We note that a similar MIQCP-based approach was implemented by the authors in preliminary experiments which did not yield competitive results and can be found along with the source code.

References

1 Tobias Achterberg, Robert E Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 32(2):473–506, 2020. doi:10.1287/IJOC.2018.0857.

2 Antonios Antoniadis, Mark de Berg, Sándor Kisfaludi-Bak, and Antonis Skarlatos. Computing smallest convex intersecting polygons. *Journal of Computational Geometry*, 16(1):167–202, 2025. doi:10.20382/jocg.v16i1a6.

3 David L. Applegate, Robert E. Bixby, Vasek Chvátal, William J. Cook, Daniel G. Espinoza, Marcos Goycoolea, and Keld Helsgaun. Certification of an optimal TSP tour through 85,900 cities. *Oper. Res. Lett.*, 37(1):11–15, 2009. doi:10.1016/J.ORL.2008.09.006.

4 Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. The lawnmower problem. In *Canadian Conference on Computational Geometry (CCCG)*, pages 461–466, 1993. URL: https://cglab.ca/~cccg/proceedings/1993/Paper79.pdf.

5 Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17:25–50, 2000. doi:10.1016/S0925-7721(00)00015-8.

6 Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

7 Armin Biere, Matti Järvisalo, and Benjamin Kiesl. Preprocessing in SAT solving. In *Handbook of Satisfiability*, pages 391–435. IOS Press, 2021. doi:10.3233/FAIA200992.

8 S.P. Boyd and L. Vandenberghe. *Convex Optimization, Pt. 1*. Berichte über verteilte Messsysteme. Cambridge University Press, 2004. URL: https://books.google.de/books?id=mYm0bLd3fcoC.

9 Francesco Carrabs, Carmine Cerrone, Raffaele Cerulli, and Manlio Gaudioso. A novel discretization scheme for the Close Enough Traveling Salesman Problem. *Comput. Oper. Res.*, 78:163–171, 2017. doi:10.1016/J.COR.2016.09.003.

10 Bernard Chazelle and David P. Dobkin. Optimal convex decompositions. In Godfried T. Toussaint, editor, *Computational Geometry*, volume 2 of *Machine Intelligence and Pattern Recognition*, pages 63–133. North-Holland, 1985. doi:10.1016/B978-0-444-87806-9.50009-8.

11 Andrew Clark. A submodular optimization approach to the metric Traveling Salesman Problem with neighborhoods. In *IEEE Conference on Decision and Control, CDC*, pages 3383–3390, 2019. doi:10.1109/CDC40024.2019.9030031.

12 William J Cook, David L Applegate, Robert E Bixby, and Vasek Chvatal. *The Traveling Salesman Problem: A computational study*. Princeton University Press, 2011.

13 Walton Pereira Coutinho, Roberto Quirino do Nascimento, Artur Alves Pessoa, and Anand Subramanian. A branch-and-bound algorithm for the Close-Enough Traveling Salesman Problem. *INFORMS J. Comput.*, 28(4):752–765, 2016. doi:10.1287/IJOC.2016.0711.

14 Mark de Berg, Joachim Gudmundsson, Matthew J. Katz, Christos Levcopoulos, Mark H. Overmars, and A. Frank van der Stappen. TSP with neighborhoods of varying size. *J. Algorithms*, 57(1):22–36, 2005. doi:10.1016/J.JALGOR.2005.01.010.

15 Jindřiška Deckerová, Kristýna Kučerová, and Jan Faigl. On improvement heuristic to solutions of the close enough traveling salesman problem in environments with obstacles. In *2023 European Conference on Mobile Robots (ECMR)*, pages 1–6, 2023. doi:10.1109/ECMR59166.2023.10256328.

16 Jindřiška Deckerová, Petr Váňa, and Jan Faigl. Combinatorial lower bounds for the generalized traveling salesman problem with neighborhoods. *Expert Systems with Applications*, 258:125185, 2024. doi:10.1016/j.eswa.2024.125185.

17 Erik D Demaine, Sándor P Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing convex partitions for point sets in the plane: The CG:SHOP Challenge 2020. *arXiv preprint arXiv:2004.04207*, 2020. arXiv:2004.04207.

18 Erik D Demaine, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Area-optimal simple polygonalizations: The CG Challenge 2019. *ACM Journal on Experimental Algorithms*, 27(2):1–12, 2022.

19 Andrea Di Placido, Claudia Archetti, Carmine Cerrone, and Bruce Golden. The generalized Close Enough Traveling Salesman Problem. *European Journal of Operational Research*, 310(3):974–991, 2023. doi:10.1016/j.ejor.2023.04.010.

20 Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph S. B. Mitchell. Touring a sequence of polygons. In *Symposium on Theory of Computing (STOC)*, pages 473–482, 2003. doi:10.1145/780542.780612.

21 Adrian Dumitrescu and Joseph SB Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135–159, 2003. doi:10.1016/S0196-6774(03)00047-6.

22 Günther Eder, Martin Held, Steinþór Jasonarson, Philipp Mayer, and Peter Palfrader. Salzburg database of polygonal data: Polygons and their generators. *Data in Brief*, 31:105984, 2020. doi:10.1016/j.dib.2020.105984.

23 Khaled M. Elbassioni, Aleksei V. Fishkin, and René Sitters. On approximating the TSP with intersecting neighborhoods. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 213–222, 2006. doi:10.1007/11940128_23.

24 Jan Faigl. Gsoa: Growing self-organizing array – unsupervised learning for the close-enough traveling salesman problem and other routing problems. *Neurocomputing*, 312:120–134, 2018. doi:10.1016/j.neucom.2018.05.079.

25 Jan Faigl and Libor Přeučil. Self-organizing map for the multi-goal path planning with polygonal goals. In Timo Honkela, Włodzisław Duch, Mark Girolami, and Samuel Kaski, editors, *International Conference on Artificial Neural Networks and Machine Learning (ICANN)*, pages 85–92, 2011.

26 Sándor P. Fekete, Dominik Krupke, Michael Perk, Christian Rieck, and Christian Scheffer. A closer cut: Computing near-optimal tours for the Lawn Mowing Problem. In *Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 1–14, 2023. doi:10.1137/1.9781611977561.

27 Sándor P. Fekete, Dominik Krupke, Michael Perk, Christian Rieck, and Christian Scheffer. The Lawn Mowing Problem: From algebra to algorithms. In *European Symposium on Algorithms (ESA)*, pages 45:1–45:18, 2023. doi:10.4230/LIPIcs.ESA.2023.45.

28 Barend Gehrels, Bruno Lalande, Mateusz Loskot, Adam Wulkiewicz, Menelaos Karavelas, and Fisikopoulos. Vissarion. Boost geometry 1.83, 2024. URL: https://www.boost.org/.

29 Iacopo Gentilini, François Margot, and Kenji Shimada. The Travelling Salesman Problem with neighbourhoods: MINLP solution. *Optimization Methods Software*, 28(2):364–378, 2013. doi:10.1080/10556788.2011.648932.

30 Daniel H Greene. The decomposition of polygons into convex parts. *Computational Geometry*, 1:235–259, 1983.

31 Gurobi Optimization LLC. Gurobi Optimizer 12.0. https://www.gurobi.com/, 2025.

32 Keld Helsgaun. Solving the equality generalized Traveling Salesman Problem using the Lin–Kernighan–Helsgaun algorithm. *Mathematical Programming Computation*, 7(3):269–287, September 2015. doi:10.1007/s12532-015-0080-8.

33 IBM. IBM ILOG CPLEX Optimization Studio. https://www.ibm.com/products/ilog-cplex-optimization-studio, 2025.

34 Dae-Sung Jang, Hyeok-Joo Chae, and Han-Lim Choi. Optimal control-based uav path planning with dynamically-constrained tsp with neighborhoods. In *International Conference on Control, Automation and Systems (ICCAS)*, pages 373–378, 2017.

35 Rouven Kniep, Dominik Krupke, and Michael Perk. tubs-alg/TSPN-SoCG-2026. Software, swhId: swh:1:dir:798ae47a3445f53c64575fac631a25a1921a137b (visited on 2026-05-15). URL: https://github.com/tubs-alg/TSPN-SoCG-2026, doi:10.4230/artifacts.26083.

36 Gilbert Laporte, Ardavan Asef-Vaziri, and Chelliah Sriskandarajah. Some applications of the generalized Travelling Salesman Problem. *Journal of the Operational Research Society*, 47(12):1461–1467, 1996. doi:10.1057/jors.1996.190.

37 Andrzej Lingas. The power of non-rectilinear holes. In *International Colloquium on Automata, Languages and Programming (ISAAC)*, pages 369–383, 1982. doi:10.1007/BFB0012784.

38 Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.

39 Joseph S. B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. In *Symposium on Discrete Algorithms (SODA)*, pages 11–18, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283385.

40 OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

41 G. Reinelt. TSPLIB–a Traveling Salesman Problem library. *ORSA Journal of Computing*, 3(4):376–384, 1991. doi:10.1287/IJOC.3.4.376.

42 Stephen L. Smith and Frank Imeson. Glns: An effective large neighborhood search heuristic for the generalized Traveling Salesman Problem. *Computers & Operations Research*, 87:1–19, 2017. doi:10.1016/j.cor.2017.05.010.

43 Xuehou Tan and Bo Jiang. Efficient algorithms for touring a sequence of convex polygons and related problems. In *Theory and Applications of Models of Computation (TAMC)*, volume 10185, pages 614–627, 2017. doi:10.1007/978-3-319-55911-7_44.

44 The CGAL Project. CGAL 6.0. https://www.cgal.org, 2025.

45 Bo Yuan and Tiantian Zhang. Towards solving TSPN with arbitrary neighborhoods: A hybrid solution. In *Artificial Life and Computational Intelligence (ACALCI)*, pages 204–215, 2017. doi:10.1007/978-3-319-51691-2_18.

Computing the Skyscraper Invariant

Marc Fersztand
University of Oxford, Oxford, United Kingdom

Jan Jendrysiak
TU Graz, Graz, Austria

Abstract

We develop the first algorithms for computing the Skyscraper Invariant [FJNT24]. This is a filtration of the classical rank invariant for multiparameter persistence modules defined by the Harder-Narasimhan filtrations along every central charge supported at a single parameter value.

Cheng's algorithm [Cheng24] can be used to compute HN filtrations of arbitrary acyclic quiver representations in polynomial time in the total dimension, but in practice, the large dimension of persistence modules makes this direct approach infeasible. We show that by exploiting the additivity of the HN filtration and the special central charges, one can get away with a brute-force approach. For d-parameter modules, this produces an FPT ε-approximate algorithm with runtime dominated by $\mathcal{O}(1/\varepsilon^d \cdot T_{\text{DEC}})$, where T_{DEC} is the time for decomposition, which we compute with AIDA [DJK25].

We show that the wall-and-chamber structure of the module can be computed via lower envelopes of degree $d-1$ polynomials. This allows for an exact computation of the Skyscraper Invariant roughly in $\mathcal{O}(n^d \cdot T_{\text{DEC}})$ time for n the size of the presentation and enables a fast hybrid algorithm.

For 2-parameter modules, we have implemented not only our algorithms but also, for the first time, Cheng's algorithm. We compare all algorithms and, as a proof of concept for data analysis, compute a filtered version of the Multiparameter Landscape for biomedical data.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Computing methodologies → Algebraic algorithms

Keywords and phrases Topological Data Analysis, Multiparameter Persistence, Persistence, Harder-Narasimhan Filtration, Skyscraper Invariant

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.47

Related Version *Full Version*: `https://arxiv.org/abs/2603.23560` [28]

Supplementary Material
Software (Source Code for Cheng's algorithm): `https://github.com/marcf-ox/sky-inv-quiv` [25]
archived at `swh:1:dir:b620e11fefcb30adcafaf299484f00c1b0a2af70`
Software (Source Code for computing the Skyscraper invariant): `https://github.com/JanJend/Skyscraper-Invariant` [38]
archived at `swh:1:dir:8ed2562604075467f0b0014975100a2d26e434ba`

Funding *Marc Fersztand*: Member of the Centre for Topological Data Analysis, funded by the EPSRC grant EP/R018472/1.
Jan Jendrysiak: Supported by Austrian Science Fund (FWF) grant P 33765-N and W1230.

Acknowledgements We are indebted to Vidit Nanda who invited the second author to visit the University of Oxford and instigated our collaboration during this visit. We also thank Ulrike Tillmann and Michael Kerber for their support, supervision, and helpful discussions.

Introduction. Multiparameter Persistence Modules (MPM) provide a way to capture topological features of multifiltered data. Unlike the one parameter case, the decomposition of an MPM is, in general, neither easy to interpret nor robust to noise [4]. To use MPM in practice, one needs invariants that are informative, stable, computable, and easy either to



© Marc Fersztand and Jan Jendrysiak;
licensed under Creative Commons License CC-BY 4.0



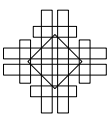




Figure 1 View from Empire State Building, Gotscho-Schleisner, 1932.

interpret or to integrate into a statistical or Machine Learning pipeline. A first example is given by the *rank invariant* [11, 12] and its vectorisations [16, 60], which have been used for applications in biology [61, 6]. In [27, 26] Fersztand, Jacquard, Tillmann, and Nanda introduced the *Skyscraper Invariant* (Figure 1), a *filtration* of the rank invariant.

Given a d-parameter module V, $\theta \in \mathbb{R}_+$ and parameters $\alpha \leq \beta \in \mathbb{R}^d$, the value $s_V^\theta(\alpha, \beta)$ of the skyscraper invariant is the rank of the structure map $V_\alpha \to V_\beta$ after – informally – restricting it to elements that persist over a parameter region whose volume is at most $1/\theta$. It is defined by the *Harder-Narasimhan* filtration for a central charge concentrated at α. It is strictly stronger than the rank invariant, carries a more discriminating erosion-distance [26], and induces filtrations of all the aforementioned vectorisations of the rank invariant. To compute an ε-approximation of the Skyscraper Invariant, we need to compute the HN filtration at each α on an ε-grid in the parameter space.

Contributions. In Section 1, we give an elementary introduction to Harder-Narasimhan (HN) filtrations and a post-hoc motivation for their utility in Persistence Theory.

For quiver representations, computing HN filtrations was recently made possible in polynomial time via Cheng's algorithm [15], described in Section 4. We present the first implementation of Cheng's algorithm using the random method introduced in [30] to compute shrunk subspaces.

The obtained algorithm computes with high probability the Skyscraper Invariant in $\mathcal{O}(\mathfrak{t}(V)^{19}\varepsilon^{-11d})$ time, up to poly-logarithmic factors, where $\mathfrak{t}(V)$, the *thickness* of V, is the maximal pointwise dimension. We optimized Cheng's algorithm to our setting, leading in Proposition 23 to an improvement of a factor $\mathfrak{t}(V)^2\varepsilon^{-2d}$ compared to the general version of the algorithm (Corollary 22). Building on ideas from [35], we propose and implement further optimisations that yield considerable empirical speed-ups.

A priori, this method is generally not directly applicable to the large data considered in TDA applications. In Section 2 we introduce a strategy which significantly cuts the computation time of the Skyscraper Invariant: Algorithm 2 reduces the computation of the HN filtration at $\alpha \in \mathbb{R}$ to *indecomposable* submodules *uniquely generated* at α. It blends the

decomposition algorithm AIDA and kernel computation with MPFREE[1].

Unfortunately, even when restricted to these small submodules, Cheng's algorithm in this form turns out to be too slow, which we prove experimentally in Table 2. Let $k \coloneqq \max_{W \subset V} \mathfrak{t}(W)$ be the maximal thickness of an indecomposable, uniquely generated submodule. We observed that for modules generated by the Persistent Homology of many typical bifiltrations, k is always 1 when the underlying point set is 2-dimensional (Conjecture 10). Otherwise, k is rarely larger than 3.

This low dimensionality enables the computation of the HN filtration with a brute-force method, which we describe and improve in Section 3. Together with Algorithm 2, this leads to an exhaustive search algorithm to compute the Skyscraper Invariant up to an ε error. Its runtime is fixed-parameter tractable (FPT) in k: $\mathcal{O}\left(1/\varepsilon^d \left(T_{\text{DEC}}(V) + \mathfrak{t}(V) 2^{k^2+\mathcal{O}(k)}\right)\right)$, where T_{DEC} is the time needed for decomposition.

The bottleneck of this computation is the iteration over all α in an ε-grid, resulting in the $1/\varepsilon^d$-factor in the computation time. To avoid this, we investigate, in Section 5, how the HN filtration at $\alpha \in \mathbb{R}^d$ changes when α varies. We define an equivalence relation $\sim_{\text{HN}(V)}$ on $\mathbb{R}^d$ by requiring the HN filtrations of V at two equivalent degrees to differ only by an update of the degrees of the generators and relations of their factors [28, Definition 6.13].

▶ **Theorem 1.** *Let V be f.p., have bounded support, and let $\bigcup_{j \in J} C_j$ be the tiling of $\mathbb{R}^d$ induced by the Betti-numbers of V. The partition induced by $\sim_{HN(V)}$ on every C_j is given by the minimisation diagram of a finite set of multilinear polynomials of degree $d-1$.*

This partition is a slice of the *wall-and-chambers* structure (e.g. [32]).

For 2-parameter modules, the equivalence classes are convex polygons and easy to compute using Algorithm 5. The full procedure, [28, Algorithm 7], creates a data structure from which the value of the Skyscraper Invariant can be queried in average time $\log(n) + \mathfrak{t}(V) \log \mathfrak{t}(V)$, where n is the number of generators and relations of a minimal presentation of V.

Our final algorithm, the *parallel grid scan*, blends this exact computation with the approximation to avoid the $(1/\varepsilon^d)$-bottleneck (Figure 13). We have implemented it and use it to demonstrate the applicability of the Skyscraper Invariant in data analysis by computing filtered Multiparameter Landscapes for biological data from [61].

Related Work. Many informative invariants of MPM have been proposed in the literature, including, among others, [9, 8, 41, 48, 53, 2], and for some, there are also implemented algorithms [21, 60, 16, 13, 47, 45, 62]. All of these are either not stronger than the rank invariant, not (yet) practically computable for large modules, or unstable in interleaving distance, demonstrating the advantage of our work. The idea of computing filtrations has also appeared in [7, 52]. An algorithm for the Skyscraper Invariant has been implemented in the special case of ladder persistence modules [37]. The computation of HN filtrations of quiver representations is an active topic in complexity theory [35, 30].

1 The Skyscraper Invariant for Multiparameter Persistence Modules

Multiparameter Persistence. Topological Data Analysis seeks to compute topological invariants from geometric data. In a typical setting, one constructs a simplicial complex X filtered by a map $f\colon X \to \mathbb{R}^d$. The sublevel sets $X_{\leq \alpha} \coloneqq \{x \in X \mid f(x) \leq \alpha\}$ assemble to a

[1] `https://github.com/JanJend/AIDA`, `https://bitbucket.org/mkerber/mpfree/`

functor $X_{\leq \bullet}\colon \mathbb{R}^d \to \mathbf{SCpx}$ via inclusion. Composition with homology $\mathbf{H}_*(-;\mathbf{k})$ over a fixed field $\mathbf{k}$ yields a functor $V\colon \mathbb{R}^d \to \mathbf{Vect}_\mathbf{k}$ - a *Multiparameter Persistence Module*.

The most important use case in practice are *density-scale* bifiltrations:

▶ **Example 2.** A standard 1-parameter pipeline [22] on a point set looks as follows: One constructs the Vietoris-Rips complex, filtered by one parameter, and computes its first persistent homology group, which decomposes into a barcode or persistence diagram (Figure 2).

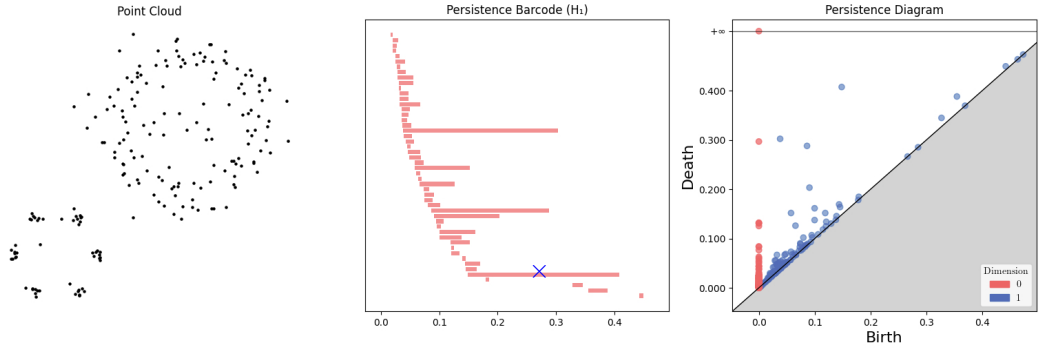


Figure 2 The lower left circle produces the bar marked with the blue cross.

The large circle on the right is impossible to detect with this technique without knowing the intensity of the noise we need to remove. Instead, we filter the simplicial complex again.

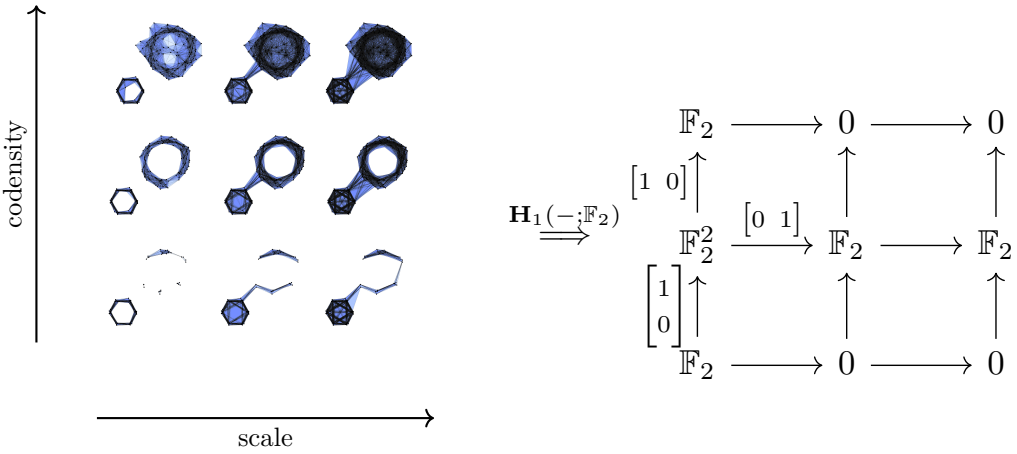


Figure 3 A 3×3 Density-Rips bifiltration of Figure 2 and its first homology group over $\mathbb{F}_2$.

Intervals. The analog of a *bar* in the multiparameter case is the *Interval Module*.

▶ **Definition 3** (Interval Module)**.** *A sub poset $I \subset \mathbb{R}^d$ is an* interval *if, with respect to the order, it is convex and connected. The module $\mathbf{k}[I]\colon \mathbb{R}^d \to \mathbf{Vect}_\mathbf{k}$ is defined as*

$$\mathbf{k}[I]_\alpha = \begin{cases} \mathbf{k} & \text{if } \alpha \in I \\ 0 & \text{otherwise} \end{cases} \qquad \mathbf{k}[I]_{\alpha\to\beta} = \begin{cases} \mathrm{Id}_\mathbf{k} & \text{if } \alpha, \beta \in I \\ 0 & \text{otherwise} \end{cases}.$$

An interval module *is a persistence module that is isomorphic to $\mathbf{k}[I]$ for some interval I.*

Motivation. The Skyscraper Invariant arises from the *Harder-Narasimhan filtration* [31] of persistence modules for a specific set of stability conditions. We will explain how one *could* arrive at considering HN filtrations even without prior knowledge of geometric invariant theory [54]. This theory was originally developed for persistence modules in [27] and [26].

Lifetime. Let V be a *multiparameter* persistence module. A sensible idea to extend the barcode is the following: search for an element with a "maximal lifetime" (an idea developed further in [52]). By "lifetime", we mean for any element $v \in V_\alpha$, with $\alpha \in \mathbb{R}^d$, the set

$$L_v := \{\beta \in \mathbb{R}^d \mid \alpha \leq \beta \text{ and } V_{\alpha\to\beta}(v) \neq 0\}$$

of parameter values under which v persists and "maximal" refers here to its volume. To assure that every lifetime has a finite volume, we will assume that every module is bounded, which can always be realised by restricting the module to a bounded subset of $\mathbb{R}^d$.

▶ **Example 4.** Consider again the bifiltered simplicial complex $\mathrm{VR}(X)\colon \mathbb{R}^2 \to \mathbf{SCpx}$ obtained from the point set X from Figure 2 filtered by a kernel density estimate. The 2-parameter module produced by $\mathbf{H}_1(\mathrm{VR}(X), \mathbb{F}_2)$ can be visualized by its *Hilbert function* - the pointwise dimension, which we denote by $\underline{\dim}_{\mathbf{k}}\, \mathbf{H}_1(\mathrm{VR}(X), \mathbb{F}_2)\colon \mathbb{R}^d \to \mathbb{N}$.

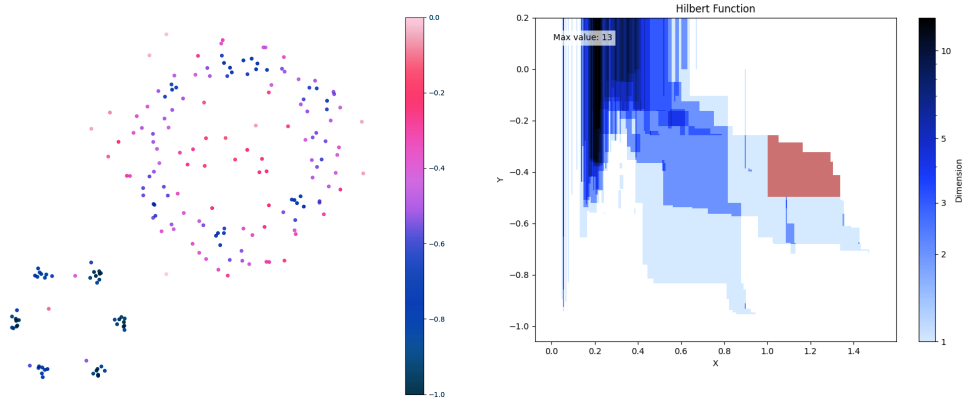


Figure 4 X coloured by density estimate and the Hilbert function of the module $\mathbf{H}_1(\mathrm{VR}(X), \mathbb{F}_2)$.

There is a non-zero vector at $\alpha = (1.0, -0.5)$ corresponding to the large circle in $\mathrm{VR}(X)_{1.0}$ after filtering out points of density lower than 0.5. We mark its lifetime red in Figure 4.

After finding the vector v of maximal lifetime, the submodule $\langle v\rangle$ of V generated by v is an interval module and L_v is the support $\mathrm{supp}\langle v\rangle$ of $\langle v\rangle$. Hence, if we compute the quotient $V/\langle v\rangle$, and repeat this process, we obtain a filtration of V in terms of interval modules.

Additivity. The element of maximal lifetime is not compatible with direct sums of modules.

▶ **Example 5.** Consider now Figure 5, where we return to the module from Figure 3. It can be split into two indecomposable summands, each roughly corresponding to one circle in the point set. The element of maximal lifetime is the vector $(1,1)^t \in \mathbb{F}_2^2$.

If we instead consider the *minimal* lifetime of an element, the construction becomes additive, meaning the desired vector lies in a direct summand:

$$\text{If } v \in V,\ w \in W, \text{ then for } v + w \in V \oplus W\colon \quad L_{v+w} = L_v \cup L_w. \tag{1}$$

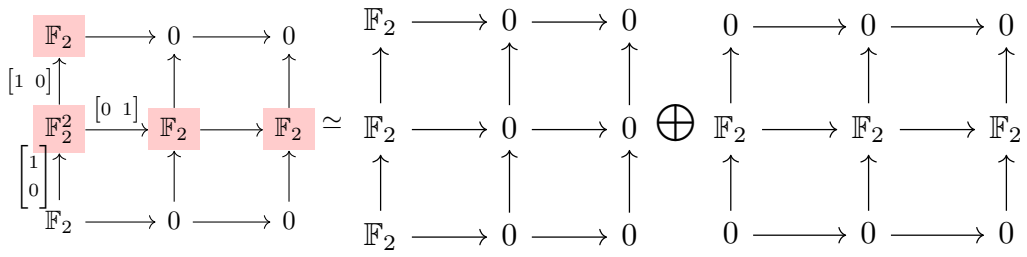


Figure 5 The maximal lifetime in the left module is indicated by red boxes.

This enables the use of decomposition algorithms (e.g. [20]) to speed up the search. Now, a priori, this element is not well defined, since in every non-zero persistence module we can find a sequence of elements whose lifetime approaches a set of empty volume. To fix this, we choose a concrete parameter value $\alpha \in \mathbb{R}^d$ and demand that the element be in V_α.

▶ **Definition 6.** *Let $S \subset V$ be a subset of V. We denote by $\langle S \rangle \subset V$ its* induced submodule.

Since we are only considering elements at α, we can now restrict our computations to $\langle V_\alpha \rangle$ – and this module is simpler than V:

▶ **Definition 7.** *V is* uniquely generated *if it has a set of generators of the same degree.*

In particular, $\langle V_\alpha \rangle$ is more decomposable than V. Still, the choice of an element is not unique: there could be multiple vectors whose lifetime has the same (minimal) volume. This is the first hint that one should not only consider single elements in V_α, but instead consider *submodules* of arbitrary dimension generated at a parameter value α.

Stability. Let $U \subset V_\alpha$ be any sub vector space. The integral of the Hilbert function $\underline{\dim}\langle U \rangle$ over $\mathbb{R}^d$ would be the natural generalisation of the volume of the lifetime. However, using this definition would never make a subvector space of dimension higher than 1 be the candidate for minimal lifetime. We need to average over the dimension of U.

▶ **Definition 8.** *Let $U \subset V_\alpha$. The* slope *of $\langle U \rangle$ (at α) is defined as*

$$\mu(\langle U \rangle) \coloneqq \frac{\dim_{\mathbf{k}} U}{\int_{\mathbb{R}^d} \underline{\dim}_{\mathbf{k}} \langle U \rangle} = \frac{\dim_{\mathbf{k}} \langle U \rangle_\alpha}{\int_{\mathbb{R}^d} \underline{\dim}_{\mathbf{k}} \langle U \rangle}.$$

By replacing, in the last term, $\langle U \rangle$ with any *persistence module we get a general definition of the slope at α. A module is called* semistable *[54, 42, 10] if no submodule has a higher slope.*

The maximal submodule of highest slope $\langle U_1 \rangle \subset \langle V_\alpha \rangle$ is semistable. We define $\langle U_2 \rangle$ as the maximal submodule of highest slope in $\langle V_\alpha \rangle / \langle U_1 \rangle$ and continue iteratively. This produces a sequence of semistable modules $(\langle U_i \rangle)_{i \in [\ell]}$. By defining inductively $F^i \coloneqq U_i + F^{i-1}$, we arrive at the *Harder-Narasimhan-filtration* [31]

$$0 = \langle F^0 \rangle \subsetneq \langle F^1 \rangle \subsetneq \cdots \subsetneq \langle F^\ell \rangle = \langle V_\alpha \rangle \tag{2}$$

of $\langle V_\alpha \rangle$ by [26, Theorem (A)]. The semistable modules $\langle U_i \rangle$ are by definition isomorphic to the factors $\langle F^i \rangle / \langle F^{i-1} \rangle$. The sequence $\mu(\langle U^i \rangle)$ of slopes is decreasing in i, but this is not obvious. For a formal treatment, see [26] and [28, §2.4].

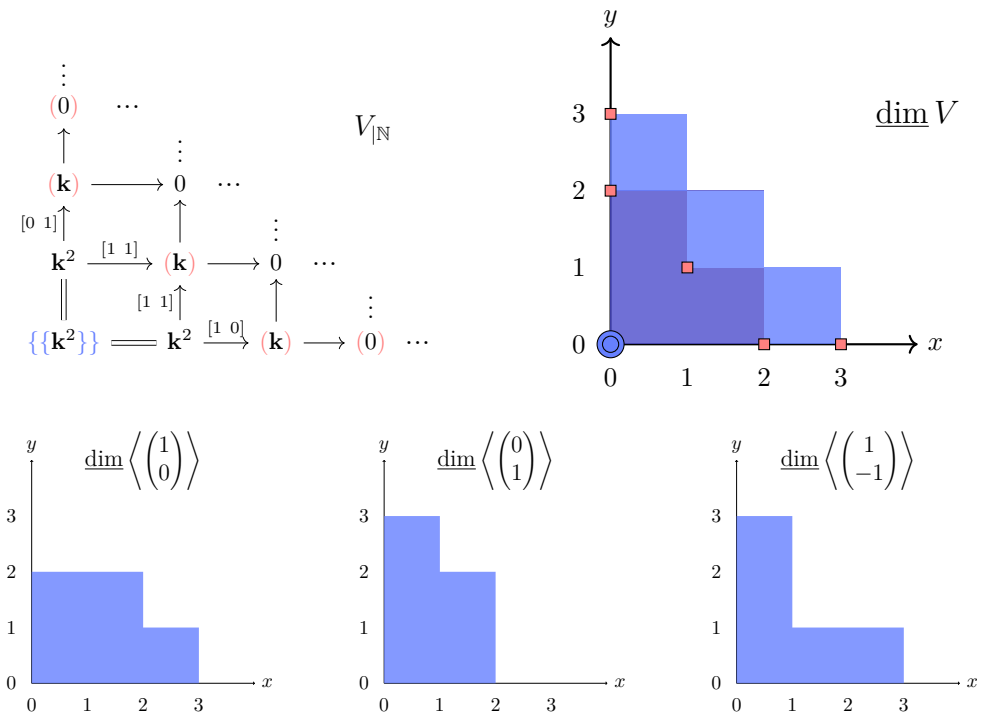


Figure 6 A module, its dimension over $\mathbb{R}^2$, and candidates for submodules of high slope.

▶ **Example 9.** In Figure 6, we construct a semistable module V that is not an interval.

At $\alpha := (0,0)$, each of the three shown submodules has slope $1/5$. Every other dimension 1 subspace has a lifetime equal to the whole support of the module and therefore slope $1/6$. The whole module on the other hand has slope $2/9$, so it is semistable.

Because of the existence of modules like Example 9, the factors of the Harder-Narasimhan filtration at α should generally not all be intervals. Surprisingly, they did turn out to be intervals for all of our examples of density-scale bifiltrations of point sets in $\mathbb{R}^2$.

▶ **Conjecture 10.** *Let $S \subset \mathbb{R}^2$ be a finite point set and $X_\bullet$ be any density-scale bifiltration of S. All indecomposable uniquely generated submodules of $\mathbf{H}_1(X_\bullet, \mathbf{k})$ are intervals.*

Then, in particular, all factors of the Harder-Narasimhan filtration are intervals by additivity.

The Skyscraper Invariant. Let $0 \hookrightarrow F^1 \hookrightarrow \ldots \hookrightarrow F^\ell = V_\alpha$ be the HN filtration at α.

▶ **Definition 11.** *[26] For $\alpha \leq \beta \in \mathbb{R}^d$ and $\theta \in \mathbb{R}$, the Skyscraper Invariant is*

$$s_V^\theta(\alpha,\beta) := \dim_{\mathbf{k}} \langle F^i \rangle_\beta = \sum_{j=1}^{i} \dim_{\mathbf{k}} \langle U_j \rangle_\beta \qquad \text{where } i \text{ is maximal with } \mu(U^i) \geq \theta.$$

This value only depends on the Hilbert function of the factors $\underline{\dim}\langle U_j \rangle$. Since $s_V^0(\alpha,\beta) = \dim_{\mathbf{k}} \langle V_\alpha \rangle_\beta = \operatorname{rank}(V_{\alpha \to \beta})$, the Skyscraper Invariant filters the rank invariant.

▶ **Example 12.** The modules V and W cannot be distinguished by the rank invariant.

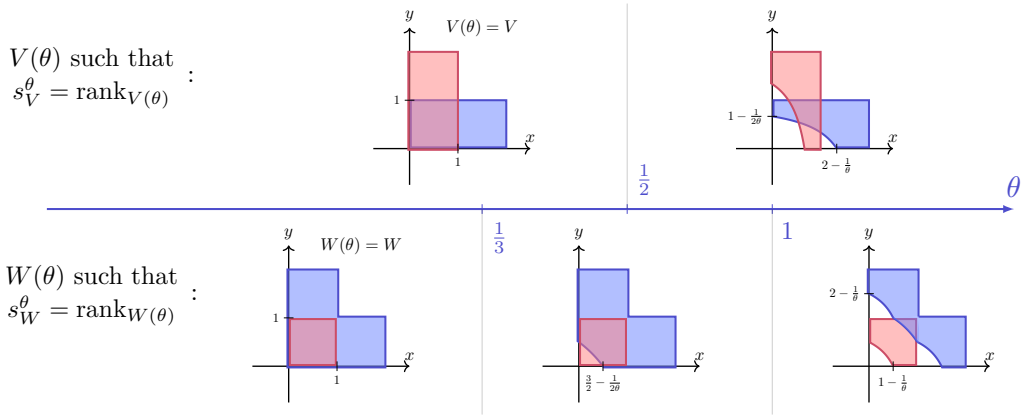


Figure 7 We visualise the Skyscraper Invariant by modules whose rank invariant is $s^\theta(\bullet, \bullet)$.

Distances. For each $\theta \in \mathbb{R}$, $s_V^\theta(\bullet, \bullet)$ is a functor from $(\mathbb{R}^d)^{\mathrm{op}} \times \mathbb{R}^d$ to $(\mathbb{Z} \cup \{+\infty\})^{\mathrm{op}}$. This allows for an *erosion distance* d_{E} [56, 55] under which the Skyscraper Invariant is stable [26, Theorem 3.7] with respect to the interleaving distance [14].

Approximation. Given $s_V^\theta(\alpha, \beta)$ for all α, β in a grid $\mathcal{G}$, we can define the functor $\widehat{s}_V^\theta(\bullet, \bullet)$ over $(\mathbb{R}^d)^{\mathrm{op}} \times \mathbb{R}^d$ via a projection to the next lower element in $\mathcal{G}$.

▶ **Lemma 13** ([28, Lemma 3.4]). *If $\mathcal{G}$ ε-approximates* $\mathrm{supp}(V)$*, then* $d_E(s, \widehat{s}) \leq \varepsilon$.

Intervals and Data Structures. After computing the HN filtrations, we need to store the Hilbert functions of the filtration factors and their associated slopes. The Hilbert functions of each factor $\langle U_i \rangle$ can be stored as a list of $\dim_{\mathbf{k}} U_i$ *intervals*, $\{W_{i,j}\}_{j \in J}$, each being assigned the slope of $\langle U_i \rangle$. For each $\alpha \in \mathcal{G}$, this results in a list of at most $\mathfrak{t}(V)$ intervals. To compute $s_V^\theta(\alpha, \beta)$ from this list, we count how many $(W_{i,j})$ with an associated slope greater than θ contain β. The latter can be done in $\mathcal{O}(\log(m))$ time where m is the number of relations of the interval. Since $m \leq n$, This puts the query time in $\mathcal{O}(\mathfrak{t}(V) \log(n))$.

Presentations. The data structure in which persistence modules come is that of a *presentation*. This is a set of generators and a set of relations among the generators. They are collected in a matrix where each row corresponds to a generator and each column to a relation. The rows and columns are endowed with the parameter value – or *degree* – of the corresponding generator or relation, making this a *graded matrix* or *monomial matrix* [49]. If we write A^G for the free module generated by a tuple of degrees G, then a graded matrix is nothing but a linear map $M \colon A^R \to A^G$ and it presents V iff $V \xrightarrow{\sim} A^G / \mathrm{Im}(M)$.

Each finitely encoded persistence modules [51] has a finite minimal presentation which can be extended to a *minimal resolution*. The degrees of the generators, relations, higher relations etc. in a minimal resolution are independent of the chosen resolution and form multi-subsets $b_0(V)$, $b_1(V), \ldots,$ $b_d(V) \subset \mathbb{R}^d$ – the *graded Betti numbers* of V. We denote the category of bounded finitely presented persistence modules by $\mathrm{Pers}_{\mathrm{fp}}^{\mathrm{b}}(\mathbb{R}^d)$.

2 Computing the Skyscraper Invariant Approximately

Our definition of the HN filtration given in the previous section is actually not consistent with the classical one [31, 42]. Instead of a filtration of $\langle V_\alpha \rangle$ one really considers a filtration of V. Only by the observation [26, Proposition 3.3] it actually restricts to one of $\langle V_\alpha \rangle$. The first simplification in our algorithms is the computation of this submodule $\langle V_\alpha \rangle$.

Induced Submodules. Computing a presentation for any induced submodule can be done by computing the kernel of a graded matrix in the category of persistence modules. For $d = 2$, this can be done with the LWKR-algorithm (Lesnick, Wright, Kerber, and Rolle [43, 40]) and for $d = 3$ there is an unpublished algorithm [5] by Bender, Gäfvert, and Lesnick. When d is larger, we rely on classical machinery, like Schreyer's algorithm [23], which is too slow in practice. We will therefore focus on the 2-parameter case for this paper.

Algorithm 1 `SubmoduleGeneration` (cf. [1, Section 7.2]).

Input: A graded matrix $M \in \mathbf{k}^{G\times R}$ presenting V and a finite subset $S \subset V$
Output: A presentation $N \in \mathbf{k}^{\deg(S)\times R_S}$ of $\langle S\rangle$

For each element in S find coordinates in G to get a graded matrix $M_S \in \mathbf{k}^{G\times \deg(S)}$
Compute $\begin{bmatrix} N \\ K \end{bmatrix} := \ker [M_S \ M] \in \mathbf{k}^{(\deg(S)\cup R)\times R_S}$
return $N \in \mathbf{k}^{\deg(S)\times R_S}$

▶ **Proposition 14.** *Algorithm 1 computes a presentation of* $\langle S\rangle$.

Proof. See [28, Proposition 3.1] for a chase in the following diagram:

$$\begin{array}{ccccccc} A[R_S] & \xrightarrow{N} & A[\deg(S)] & \xrightarrow{q} & \langle S\rangle & \longrightarrow & 0 \\ {\scriptstyle -K}\downarrow & & \downarrow{\scriptstyle M_S} & & \downarrow{\scriptstyle \iota} & & \\ A[R] & \xrightarrow{M} & A[G] & \xrightarrow{p} & V & \longrightarrow & 0. \end{array}$$

◀

Approximate Computation. We condense the ideas from Section 1 into an algorithm.

Algorithm 2 ε-approximation of the Skyscraper Invariant.

Input: $\varepsilon \in \mathbb{R}$; a graded matrix $M \in \mathbf{k}^{G\times R}$ presenting $V \in \mathrm{Pers}^{\mathrm{b}}_{\mathrm{fp}}(\mathbb{R}^d)$.
Output: For each $\alpha \in \varepsilon\mathbb{Z}^d \cap \mathrm{supp}(V)$: Lists $(W_{t,j}, \mu_t)_{t\in[\ell]}$ of elements in $\mathcal{P}(\mathbb{R}^d) \times \mathbb{R}$.

Decompose V into $\bigoplus_{i\in I} V_i$;
for $\alpha \in \varepsilon\mathbb{Z}^d \cap \mathrm{supp}(V)$ **do**
  for $i \in I$ **do**
    Compute $\langle (V_i)_\alpha\rangle$ with Algorithm 1; Decompose $N_i \xrightarrow{\sim} \bigoplus_{j\in J_i} N_{i,j}$
    for $j \in J_i$ **do**
      $(U_{i,j,t},\ \mu_{i,j,t})_{t\in[\ell_{i,j}]} \leftarrow$ HN-Filtration$_\alpha$-Factors$(N_{i,j})$
  $(U_t, \mu_t)_{t\in[\ell]} \leftarrow \mathrm{SORT}\left(\{((U_{i,j,t},\ \mu_{i,j,t})_{t\in[\ell_{i,\alpha,j}]}\}_{i,j}\right)$
  $\forall t \in [\ell]$, decompose $\underline{\dim} U_t$ into intervals $\{W_{t,j}\}_{\mathcal{J}_t}$; **return** $(W_{t,j}, \mu_t)_{t\in[\ell],j\in\mathcal{J}_t}$

Correctness. Denote by $n = |G| + |R|$ the input size, $\mathfrak{t}(V)$ the thickness of V, k the maximal thickness of a uniquely generated indecomposable submodule, and by $T_{\text{HN-F}}$, T_{KER}, and T_{DEC} the times to compute the HN filtration, kernels, and decomposition for d-parameter modules.

▶ **Proposition 15.** *Algorithm 2 returns an ε-approximation of the Skyscraper Invariant in*

$$\mathcal{O}\left(1/\varepsilon^2 \left(T_{\text{KER}}(n,d) + T_{\text{DEC}}(n,d) + (\mathfrak{t}(V)/k)(T_{\text{HN-F}}(n,k,d))\right)\right) \text{ time for } d = 2.$$

Proof. Correctness follows from additivity (Equation 1) and the fact that we only need the ordered list of Hilbert functions of factors. At each step there are at most $\mathfrak{t}(V)/k$ indecomposable submodules of thickness k generated at α. ◀

3 Finding Highest-Slope Submodule

We will explain how to compute the Harder-Narasimhan filtration of $\langle V_\alpha \rangle$ with an exhaustive search. To do so, we must compute slopes. The slope $\mu_\alpha(V)$ at α is entirely determined by the Hilbert function of $\langle V_\alpha \rangle$ and thus by its Betti numbers.

▶ **Proposition 16.** *Let $V \in \mathrm{Pers}^b_{fp}(\mathbb{R}^d)$, then*

$$\int_{\mathbb{R}^d} \underline{\dim}\, V = \sum_{i=0}^{d} (-1)^{d+i} \sum_{\gamma \in b_i(V)} \prod_{j=1}^{d} \gamma_j. \tag{3}$$

Proof. Consider a minimal free resolution $0 \to F_d \to \cdots \to F_0 \to V \to 0$ which will be of maximal length d by Hilbert's Syzygy theorem. In our bounded case, it will even be exactly of length d by duality of free and injective resolutions (first proved by Miller in [50] and rediscovered in [3] for the computation of Multiparameter Persistent Cohomology). Each F_i has a basis with degrees at $b_i(V)$. By exactness, for each $\alpha \in \mathbb{R}^d$ we have

$$\dim_{\mathbf{k}} V_\alpha = \sum_{i=0}^{d} (-1)^i \dim_{\mathbf{k}} (F_i)_\alpha = \sum_{i=0}^{d} (-1)^i \sum_{\gamma \in b_i(V)} \mathbb{1}_{\{\gamma \leq \alpha\}}. \tag{4}$$

Let $B \in \mathbb{R}^d$ be large enough that $\mathrm{supp}(V) \subset \prod_{j\in[d]} (-\infty, B_j]$, then

$$\int_{\mathbb{R}^d} \underline{\dim}\, V = \int_{\prod_{j\in[d]}(-\infty,B_j]} \underline{\dim}\, V = \sum_{i=0}^{d} (-1)^i \sum_{\gamma \in b_i(V)} \prod_{j=1}^{d} (B_j - \gamma_j)\,. \tag{5}$$

As a function of B, Equation 5 is a polynomial and constant, so we can set $B = 0$. ◀

Iteration over subspaces. Instead of considering $\langle V_\alpha \rangle$, we assume without loss of generality, that V is uniquely generated at 0. Recall that $\mathfrak{t}(V) = \max \underline{\dim}\, V$, and choose any basis $\mathbf{k}^{\mathfrak{t}(V)} \xrightarrow{\sim} V_0$. We want to find the sub vector space $U \subset \mathbf{k}^{\mathfrak{t}(V)}$ for which $\langle U \rangle \subset V$ has the highest slope. An exhaustive search would involve computing $\mu(\langle U \rangle)$ with Algorithm 1 for *all* sub vector spaces $U \subset V_0$. This is clearly possible if we work over a finite field, but also in the infinite case, the finite presentation of V allows only a finite number of non-isomorphic submodules. Since all software uses finite fields, we fix a prime power q and $\mathbf{k} = \mathbb{F}_q$.

Denote by $\mathcal{P}V_0$ the set of 1-dimensional subspaces of V_0 and define $W_{\max} \in \arg\max_{W \in \mathcal{P}V_0} \mu(\langle W \rangle)$

▶ **Proposition 17.** *Let $1 < k \leq \mathfrak{t}(V)$ and define $V_k := \{W \in \mathcal{P}V_0 \mid \mu(\langle W \rangle) \geq \mu(\langle W_{max} \rangle)/k\}$. If for every k-subspace $U \subset V_0$ it holds that $\mathcal{P}U \not\subset V_k$, then $\langle W_{max} \rangle \subset V$ has a higher slope than any other submodule of thickness k.*

Proof. Consider any $U \subset V_0$ of dimension k. By assumption, there is a 1-dimensional subspace $T \in \mathcal{P}U \subset \mathcal{P}V_0$ which is not in V_k. Therefore, $\mu(\langle T \rangle) < \mu(\langle W_{\max} \rangle)/k$.

$$\mu(\langle U \rangle) = k \Big/ \int_{\mathbb{R}^d} \underline{\dim}_{\mathbf{k}} \langle U \rangle < k \Big/ \int_{\mathbb{R}^d} \underline{\dim}_{\mathbf{k}} \langle T \rangle = k\mu(\langle T \rangle) < \mu(\langle W_{\max} \rangle). \qquad ◀$$

Every k-dimensional vector space over $\mathbb{F}_q$ contains exactly $(q^k - 1)/(q - 1)$ subspaces of dimension 1, so the premise of Proposition 17 is trivially satisfied if $|V_k| < (q^k - 1)/(q - 1)$. V_0 has $\sim q^{\mathfrak{t}(V)^2/4}$ subspaces, but only $\sim q^{\mathfrak{t}(V)}$ of these are 1-dimensional.

Algorithm 3 Exhaustive search for the highest-slope submodule.

Input: A graded matrix $M \in \mathbb{F}_q^{B_0 \times B_1}$ presenting $V \in \mathrm{Pers}_{\mathrm{fp}}^{\mathrm{b}}(\mathbb{R}^d)$ generated at 0.
Output: A subspace $S \subset V_0$ such that $\langle S \rangle$ is the submodule of maximal slope.

```
slopes: ({Subsets of V_0} → ℝ) ← {}; for W ∈ PV_0 do
    Pick any v ≠ 0 ∈ W, N_0 ← Algorithm 1(M, v)
    Compute a resolution N_i ∈ k^{B_{i-1} × B_i}; b_i(V) ← B_i
    slopes.append ( v ↦ Σ_{i=0}^{d} (-1)^{d+i} Σ_{γ ∈ b_i(V)} Π_{j=1}^{d} γ_j )
v_max ← arg max slopes; V_k ← {w ∈ PV_0 | slopes(w) ≥ slopes(v_max)/k}
for 1 ≤ k ≤ t(V) do
    if |V_k| ≥ (q^k − 1)/(q − 1) then
        for W ∈ Gr_k V_0 do
            Pick a basis B ⊂ W, N_0 ← Algorithm 1(M, B)
            Compute a resolution N_i ∈ k^{B_{i-1} × B_i}; b_i(V) ← B_i
            slopes.append ( B ↦ Σ_{i=0}^{d} (-1)^{d+i} Σ_{γ ∈ b_i(V)} Π_{j=1}^{d} γ_j )
return arg max (slopes)
```

▶ **Proposition 18.** *Algorithm 3 returns the correct result in* $\mathcal{O}(T_{\mathrm{KER}} \cdot q^{\mathfrak{t}(V)^2/4+\mathcal{O}(\mathfrak{t}(V))})$ *time.*

Proof. Correctness follows from Proposition 16 and Proposition 17. Algorithm 1 computes a kernel for each subspaces of $\mathbb{F}_q^{\mathfrak{t}(V)}$, of which there are $\sum_i \binom{\mathfrak{t}(V)}{i}_q \sim q^{(\mathfrak{t}(V)/2)^2+\mathcal{O}(\mathfrak{t}(V))}$. ◀

By calling Algorithm 3 recursively on the quotient by the submodule of highest slope we can compute the whole HN filtration at α in $\mathcal{O}(\mathfrak{t}(V) \cdot T_{\mathrm{KER}} \cdot q^{\mathfrak{t}(V)^2/4+\mathcal{O}(\mathfrak{t}(V))})$ time.

4 Cheng's Algorithm

Building on the work of [19, 33], Cheng reduced the computation of HN filtrations of representations of finite acyclic quivers to the shrunk subspace problem, which can be solved in deterministic polynomial-time for large enough fields [15]. This algorithm can be directly applied to persistence modules over a finite poset [26, Remark 1.13].

It does not follow directly from Lemma 13 that computing the HN filtration after restriction to the grid produces an error on $s_V^\bullet(\bullet,\bullet)$ less than ε in the erosion distance, because this could change which submodule has the highest slope. Instead, one needs to apply the functoriality of the HN filtration [24] on an interleaving [26, Theorems 2.9 and 3.7].

Shrunk subspace problem. Fix two integers $N, N' > 0$ and a field $\mathbf{k}$. The space of $N \times N'$ matrices with coefficients in $\mathbf{k}$ is denoted $\mathbf{k}^{N \times N'}$ and we write $\otimes$ for the Kronecker product.

▶ **Definition 19** (shrunk subspace). *Let* $\mathcal{A} \subset \mathbf{k}^{N \times N'}$ *be a subspace. A subspace* $U \subset \mathbf{k}^{N'}$

is a shrunk subspace *of* $\mathcal{A}$ *it it maximizes* $\quad \dim_{\mathbf{k}} U - \dim_{\mathbf{k}} \sum_{A \in \mathcal{A}} AU.$

Observe that the being a shrunk subspace of $\mathcal{A}$ *is closed under intersections. We can thus define the* minimal shrunk subspace *of* $\mathcal{A}$.

For the purpose of this paper, we present a simple random algorithm introduced in [29, Theorem 6.7] for $\mathbf{k} = \mathbb{Q}$. We extend this algorithm to any field using [34, Lemma 5.3]

▶ **Proposition 20.** *Fix a field* $\mathbf{k}$ *and* $\eta > 0$. *Given an integer* $N > 0$ *and a subspace* $\mathcal{A} \subset \mathbf{k}^{N\times N}$ *generated by a family of* ℓ *matrices, there exists a random algorithm to compute the minimal shrunk subspace of* $\mathcal{A}$ *using at most* $C_{\mathbf{k}}\ell N^{8+\eta}$ *arithmetic operations for some constant* $C_{\mathbf{k}} > 0$.

From persistence module to shrunk subspace. Fix $W \in \mathrm{Pers}_{\mathrm{fp}}^{\mathrm{b}}(\mathbb{R}^d)$ and a number $\varepsilon > 0$. Assume that the support of W is contained in a bounded cube $B \subset \mathbb{R}^d$ and let $\mathcal{G} \coloneqq (\varepsilon\mathbb{Z}^d) \cap B$. We denote by $V \colon \mathcal{G} \to \mathbf{Vect}$ the restriction of W to $\mathcal{G}$. Using the Persistence-Algebra library one can compute the structure maps of V in time $n^3(1/\varepsilon)^d + n^2(1/\varepsilon)^{2d}$.

Let $\alpha \in \mathcal{G}$, and let $\{\alpha, \beta_1, \dots, \beta_k\} \coloneqq \{\beta \in \mathcal{G} \mid \beta \geq \alpha\}$. Define $p_0 \coloneqq \dim V_\alpha$ and $q_0 \coloneqq \sum_{i=1}^{k} \dim V_{\beta_i}$. Consider the following subspace of $\mathbf{k}^{q_0\times p_0}$:

$$\mathcal{A}_\alpha \coloneqq \left\{ \begin{pmatrix} \lambda_1 V_{\alpha\to\beta_1} \\ \vdots \\ \lambda_k V_{\alpha\to\beta_k} \end{pmatrix} \;\middle|\; \lambda \in \mathbf{k}^k \right\} \tag{6}$$

▶ **Proposition 21** ([36, Theorem 4.3])**.** *Let* $p, q > 0$ *and let* $0 = F^0 \hookrightarrow F^1 \hookrightarrow \cdots \hookrightarrow F^\ell = V_\alpha$ *be the HN filtration of* V *at* α. *Let* $i(p,q)$ *be the maximum integer* $0 \leq i \leq \ell$

$$\text{such that either } i = 0 \quad \text{or} \quad \mu(F^i/F^{i-1}) > \frac{p}{p+q}.$$

The minimal shrunk subspace of $\mathbf{k}^{p\times q} \otimes \mathcal{A}_\alpha$ *is* $\mathbf{k}^q \otimes F^{i(p,q)}$.

Cheng also observed in the above Theorem that when $(p,q) \coloneqq (p_0, q_0)$, either $\ell = 1$ or $0 < i(p_0, q_0) < \ell$ [15, Corollary 3.4]. They thus devised Algorithm 4.

Algorithm 4 HN-Cheng.

```
1 Input: A finite poset 𝒢, α ∈ 𝒢 and V : 𝒢 → Vect_k uniquely generated at α.
2 Output: The list of modules in the HN filtration of V along α
3 Let U ⊂ V_α such that k^{q_0} ⊗ U is the minimal shrunk subspace of k^{p_0×q_0} ⊗ 𝒜_α
    computed by Proposition 20 ;
4 if U ∈ {0, V_α} then
5   return {0, V};
6 if U ≠ ∅ then
7   return HN-Cheng(𝒢, α, ⟨U⟩) ∪ (U + HN-Cheng(𝒢, α, ⟨V_α/U⟩));
```

▶ **Corollary 22.** *Given* $V \in \mathrm{Pers}_{fp}^{b}(\mathbb{R}^d)$ *and* $\varepsilon > 0$, *there is a random algorithm to compute an* ε*-approximation of* s_V *in time* $\mathcal{O}(\mathfrak{t}(V)^{19}\varepsilon^{-11d})$ *up to poly-logarithmic factors.*

Proof. By [26] (as detailed in [28, Corollary 4.5]), applying for every $\alpha \in \mathcal{G}$ Algorithm 4 to the restriction of V to $\mathcal{G}$ yields an ε-approximation of s_V. By Proposition 20 with $\ell \coloneqq kp_0q_0$ and $N \coloneqq p_0q_0$, line 3 in Algorithm 4 takes $\mathcal{O}(|\mathcal{G}|(p_0q_0)^{9+\eta/2}) = \mathcal{O}(\mathfrak{t}(V)^{18+\eta}\varepsilon^{-10d-\eta})$ for any $\eta > 0$. The recursive step adds a factor $\mathfrak{t}(V)$, whence the desired complexity. ◀

Improvements to Cheng's algorithm. By harnessing the block structure of (6), we are able to improve the complexity in Proposition 20 for the subspaces appearing in Algorithm 4.

▶ **Proposition 23.** *Given $V \in \mathrm{Pers}^b_{fp}(\mathbb{R}^d)$ and $\varepsilon > 0$, there is a random algorithm to compute an ε-approximation of s_V in time $\mathcal{O}(\mathfrak{t}(V)^{17}\varepsilon^{-9d})$ up to poly-logarithmic factors.*

Further empirical improvements can be obtained by computing $i(p,q)$ in Proposition 21 for small integers $p, q > 0$ such that $\frac{p}{q} \approx \frac{p_0}{q_0}$. Indeed, the complexity of computing $F^{i(p,q)}$ is heavily dependent on the size of pq. For simplicity, assume that $\frac{p}{q} > \frac{p_0}{q_0}$. If $F^{i(p,q)} \neq 0$, then we can replace U by $F^{i(p,q)}$ in Algorithm 4. Otherwise, we know that there are no submodule $U \subset V$ such that $\mu(U) > \frac{p}{p+q}$. We keep choosing better approximations $\frac{p}{q}$ of $\frac{p_0}{q_0}$ (so with higher values of p, q) until either we find a proper submodule in the HN filtration of V at α, or we can guarantee that V is semistable at α.

Finally, given $p, q > 0$ and a matrix $A \in \mathbf{k}^{p\times q} \otimes \mathcal{A}_\alpha$, our implementation uses the block structure of A to partially reduce it: Since $A = (\lambda_{ij} V_{\alpha \to \beta_{i \bmod k}})_{ij}$ for some $\Lambda \in \mathbf{k}^{pk\times q}$, we column-reduce the first q block rows of A by reducing Λ and each $(V_{\alpha\to\beta_i})_{1\le i\le k}$.

These two empirical improvements led to significant speed-ups (see [28, Table 3]).

5 Exact Computation of the Skyscraper Invariant

The slope in relation to α. For many computations on persistence modules, it is often enough to perform them only at those parameter values that share each of their coordinates with a generator or relation (cf. the barcode template [43, 44]). We call the set of these parameters, $\mathrm{Grid}(V) \subset \mathbb{R}^d$, the *induced grid* of V. It partitions $\mathbb{R}^d$ into rectangular cells.

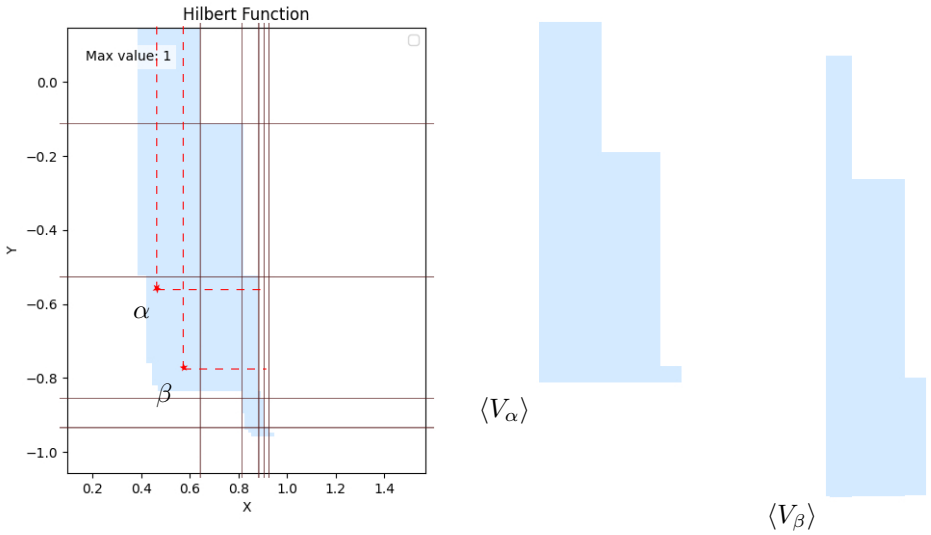


Figure 8 The direct summand corresponding to the small circle in Figure 2 with an overlaid grid.

Consider any cell C in the grid on the left in Figure 8, which is slightly coarser than the induced grid. For all $\alpha \in C$, the submodules $\langle V_\alpha \rangle$ will look qualitatively similar: Each has a presentation with the same underlying matrix. We will use this fact to avoid recomputing decompositions and HN filtrations for every $\alpha \in C$ on an ε-grid.

For $J \subset [d]$, denote by $F_J \subset \mathbb{R}^d$ the hyperplane spanned by the coordinate axes indicated by J, with the origin at α. The graded Betti numbers, which lie in some F_J for $|J| < d$, will be shifted to align with $\alpha \in C$ when α changes (Figure 9).

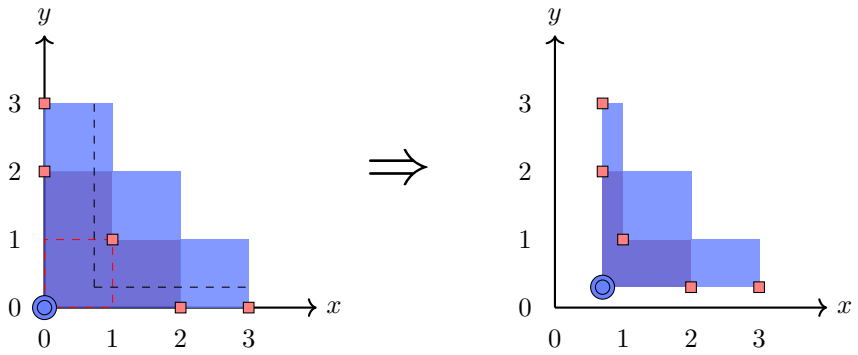


Figure 9 The uniquely generated module from Example 9 after cutting it off.

▶ **Lemma 24.** *Let V be generated at 0. For α in the first cell of Grid(V), the function $\alpha \mapsto \mu\left(\langle V_\alpha \rangle\right)^{-1}$ is a multilinear polynomial of degree d which we denote by p_V and*

$$p_V(\alpha) = \frac{1}{\mathfrak{t}(V)} \sum_{J \subset [d]} (-1)^{|J|} \int_{F_{[d]\setminus J}} \underline{\dim}\, V_{|F_{[d]\setminus J}} \alpha^J = \sum_{J \subset [d]} (-1)^{|J|} \mu\left(V_{|F_{[d]\setminus J}}\right)^{-1} \alpha^J. \tag{7}$$

In particular, the leading coefficient is $(-1)^d$ and the constant term is $\mu(V)^{-1}$.

Proof by picture: Figure 10. Full proof in [28, Lemma 6.5]

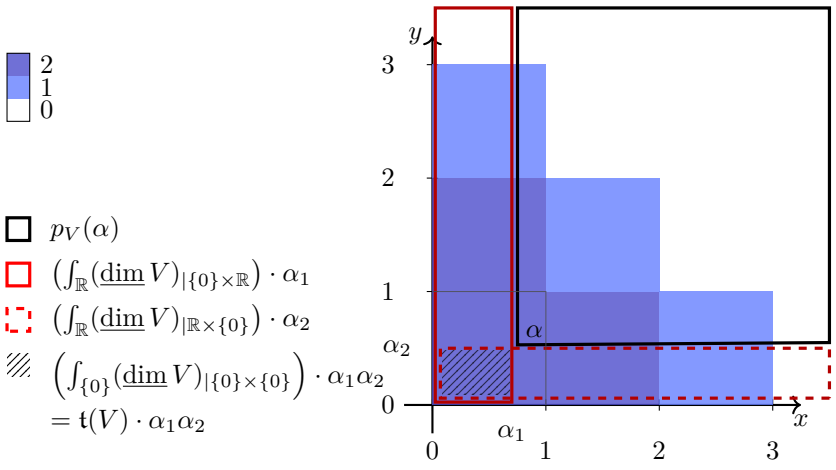


Figure 10 Equation 7 is obtained by exclusion-inclusion of the red and black rectangles. ◀.

▶ **Observation 25.** *To find the submodule of highest slope for all α in a grid cell C at the same time, compute the lower envelope of the slope polynomials of all non-isomorphic submodules. Since they have the same leading term $(-1)^d \alpha^{[d]}$ we can omit it for the computation of this diagram. For the important case of $d = 2$, this is the lower envelope of a set of planes in $\mathbb{R}^3$ and can be computed efficiently as the upper convex hull of its dual point set [18, 14.2].*

We describe this computation for $d = 2$, where after ignoring the degree-2 part, the slope-polynomials are linear and so the minimisation diagram is convex-polygonal.

Algorithm 5 Highest Slope Submodule in a Grid Cell.

Input: A graded matrix M presenting V uniquely generated at α; $C \subset \mathbb{R}^2$
Output: A convex polygonal subdivision $\{S_i\}_{i\in I}$ of C. For each $i \in I$ a sub vector space $W \subset V_\alpha$ and the truncated slope polynomial $p_{\langle W\rangle}$.

```
slope-polynomials ∈ ({Finite Subsets of V_α} × ℝ³) [ ] ← empty array
for sub vector space W ⊂ V_α do
    N ← Algorithm 1(M, B), for some basis B ⊂ W; K ∈ k^{R×S} ← ker N
    b_0(⟨W⟩) ← (deg(B)), b_1(⟨W⟩) ← R, b_2(⟨W⟩) ← S
    slope-polynomials.append(B, 1/|B| (∫(dim V)_{|F_1} x_1 + ∫(dim V)_{|F_2} x_2 + ∫ dim V))
subdivision {S_i}_{i∈I} ← LowerEnvelope( slope-polynomials)
return {(S_j ∩ C, slope-polynomials_j)}_{j∈J such that S_j∩C≠∅}
```

The thickness-6 module from Table 1 produces the following subdivision.

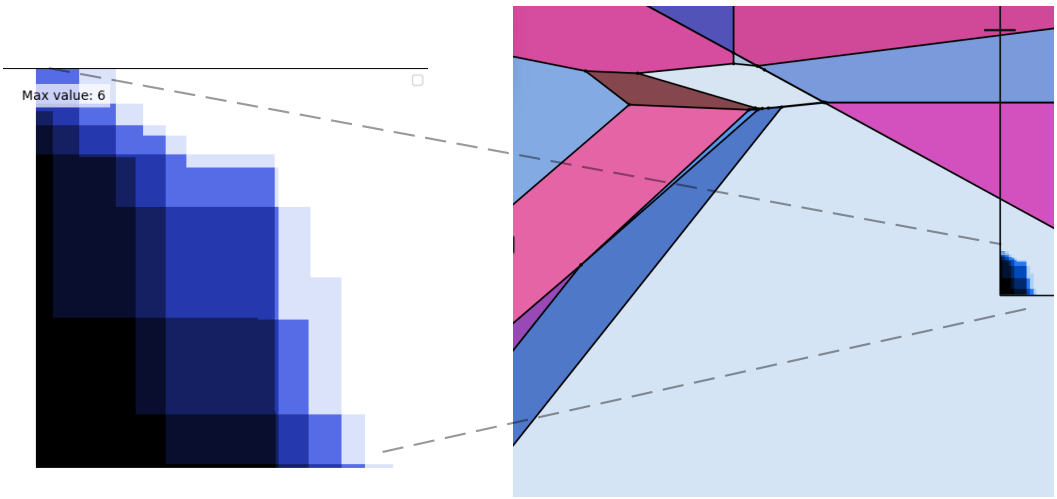


Figure 11 The minimisation diagram overlaid on the Hilbert function.

Arrangements. We ran Algorithm 5 for all modules used in Table 1. In each case, as in Figure 11, the first grid cell lies entirely in one face of the subdivision.

▶ **Proposition 26.** *Algorithm 5 is correct and runs in expected time* $\mathcal{O}(\mathfrak{t}(V)^3 2^{\mathfrak{t}(V)^2/4+\mathcal{O}(\mathfrak{t}(V))})$.

Proof. Correctness follows from Lemma 24 and Observation 25. The runtime is the same as for Algorithm 3, except that kernel computation is in $\mathfrak{t}(V)^3$ for 2-parameter modules and we compute a lower envelope. That can be done in expected time $n \log n$, putting the runtime in

$$\mathcal{O}\left(2^{\mathfrak{t}(V)^2/4+\mathfrak{t}(V)} \log(2^{\mathfrak{t}(V)^2/4+\mathcal{O}(\mathfrak{t}(V))})\right) = \mathcal{O}\left(\mathfrak{t}(V)^2 2^{\mathfrak{t}(V)^2/4+\mathcal{O}(\mathfrak{t}(V))}\right). \quad \blacktriangleleft$$

Exact Computation of the Skyscraper Invariant. For each $i \in I$, there is a highest-slope submodule $W^{\{i\}} \subset \langle V_\alpha\rangle$ associated with the face S_i. By computing the quotient $V^{\{i\}} := \langle V_\alpha\rangle / W^{\{i\}}$ and recursively calling Algorithm 5 on the pair $(V^{\{i\}}, S_i)$, we obtain a nested sequence of subdivisions of the grid cell C and an associated tree of filtration factors. Glueing these for all grid cells, we obtain a data structure from which we can compute the exact value of $s^\theta_V(\alpha, \beta)$ by locating α in the nested subdivisions [28, Section 6.4].

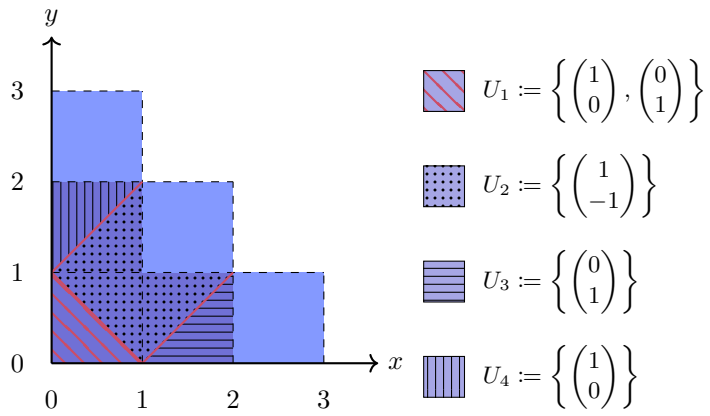


Figure 12 The wall and chamber structure of the stable module from Example 9.

Wall and Chamber Structure. For all β in a region of this nested sequence of subdivisions, the module V has equivalent HN filtrations. This is one implication of Theorem 1.

Parallel grid scanning: The final algorithm. In practice, where one wants to compute filtered vectorisations of the Skyscraper Invariant, it is not a problem to restrict the module to an ε-grid $\mathcal{G}$ and use Algorithm 2. The real power of Theorem 1 and Algorithm 5 lies in the fact that it is a recipe to avoid recomputations: For any direct summand V_i of the input module, the induced grid $\mathrm{Grid}(V_i)$ is typically coarser than $\mathcal{G}$ in some places and finer than $\mathcal{G}$ in others. We therefore overlay the two grids to form the *restriction* or *parallel* grid and use Algorithm 5 only for those cells in each $\mathrm{Grid}(V_i)$ that intersect $\mathcal{G}$. This technique enables fast computation of the Skyscraper Invariant at high resolution.

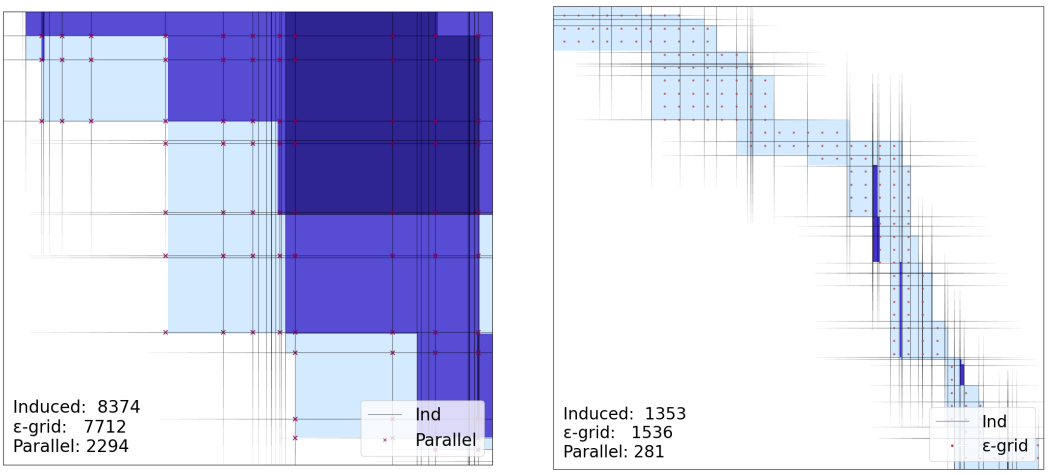


Figure 13 Enlarged detail of the Hilbert functions of $\mathbf{H}_1$ from the dataset [61] used in Section 7. Left: the induced grid in black lines overlaid on the support and the restriction grid marked with purple crosses. Right: the induced grid indicated by black lines and the ε-grid indicated by red dots. Bottom left corners: number of points in the overlaid grids - the parallel grid is much smaller.

6 Experiments

All experiments are performed on an i7-1255U Processor without any parallelisation. A timeout was set to 10 min and each run is averaged over 5 instances whenever possible. Our software and experiments are built using the PERSISTENCE-ALGEBRA C++ library[2].

Harder-Narasimhan filtrations. To generate indecomposable uniquely generated submodules of non-zero thickness we used synthetically generated noisy samples of a torus embedded in $\mathbb{R}^3$ (cf Conjecture 10) and random presentation matrices. We test the exhaustive search Algorithm 3 and its filter Proposition 17. For thickness 9, Algorithm 3 always timed out.

Table 1 Runtime (ms) of Algorithm 3. Columns indexed by dimension/thickness.

Dataset	Variant	2	3	4	5	6	7	8
Torus	with filter	2.2	2.8	5.5	19.5	202	2.7k	–
Torus	w/o filter	2.8	2.8	5.3	18.8	206	2.7k	–
Random	with filter	8.1	4.8	6.4	28.9	454	5.9k	100k
Random	w/o filter	3.6	2.3	5.1	27.1	265	3.6k	61.5k

We conclude that the filter (Proposition 17) for Algorithm 3 is barely helpful.

Runtime of Cheng's algorithm. After restricting the torus generated modules to equidistant grids, we have used Cheng's algorithm (Algorithm 4) over $\mathbb{F}_2$ and $\mathbb{Q}$. Since we can only generate modules from bifiltrations over $\mathbb{F}_2$, we interpreted them as $\mathbb{Q}$-modules when possible.

Table 2 Runtimes of Algorithm 4(ms). An asterisk means that some, and a dash that all instances timed out. Columns are indexed by dimension/thickness, and rows by grid size.

$\mathbb{Q}$ / $\mathbb{F}_2$	2	3	4	5	6	7
2x2	1.9/1.8	1.8/3.0	1.8/4.3	5.0/5.0	4.4/10.3	11.6/33.4
3x3	8.7/18.5	15.2/27.4	21.8/43.2	76.6/157k*	40.8/3.7k	53.0/301
4x4	113/2.0k	1.1k/117k*	412/223*	720/33.6k	499/66.6k	177/38.4k*
5x5	1.8k/154k*	11.8k/283*	36.8k/279k*	11.3k/–	5.2k/254k*	2.7k/–
6x6	61.5k/–	205k/–	323k/–	39.4k/–	66.6k/–	16.1k/–

We observe a great variability of runtimes for Cheng's algorithm depending on whether the heuristic introduced at the end of Section 4 successfully computes and certifies the correctness of all the HN filtrations. The random algorithm from [30] performs computations on square matrices with $\mathfrak{t}(V)^2\varepsilon^{-d}r$ rows for a large enough $r > 0$. For $\mathbf{k} = \mathbb{Q}$ (but not for $\mathbf{k} = \mathbb{F}_2$), we have observed in our experiments that $r = 1$ is large enough.

Comparison. Table 2 shows the strong dependence of Algorithm 4 on the size of the grid to which the uniquely graded module has to be restricted. Algorithm 3 does not require this restriction but, as visible in Table 1, it is highly dependent on the thickness of the input.

[2] `https://github.com/JanJend/Skyscraper-Invariant`, `https://github.com/JanJend/Persistence-Algebra`

The asymptotic runtimes of the algorithms suggest that for practical grids ($> 50 \times 50$), Cheng's algorithm would outperform the exhaustive search not before $\mathfrak{t}(V) \geq 20$. At that point, the number of arithmetic operations is too large for any computer to handle.

Runtime on practical data. We used benchmark data from [39, 61] to compare the parallel grid scan algorithm with Algorithm 2.

Table 3 Runtime (s) over a 50×50 grid on small point sets: density-alpha [63], multicover bifiltrations [59, 17] from sampled spheres and of random points. Number of points in the name.

Data	Circle	D7.5k	D15k	D30k	MC46	MC96	MC175
n	100	140	219	333	3.3k	9.1k	19.6k
Parallel	0.03	0.16	0.42	1.25	0.08	0.41	1.45
Alg 2	1.87	0.47	0.41	0.76	12.68	32.97	73.54

The parallel grid scan should not show the same quadratic dependency on the grid size of Algorithm 2. Table 4 confirms this and shows that our algorithm computes the Skyscraper Invariant in a practical timeframe for relevant data sizes.

Table 4 Runtime (s) on 40k points, uniformly sampled, and 4k locations in hypoxic2 (*grid-size*).

Data	uni 200, $\mathbf{H}_1$	uni 200, $\mathbf{H}_0$	Hyp 20	Hyp 40	Hyp 80	Hyp 160
n	113k	93.5k	9.4k	9.4k	9.4k	9.4k
Parallel	299.7	422.0	54.8	93.4	167.5	311.3

7 Application and Conclusion

Filtered Landscapes. The Multiparameter Landscape [60] is directly defined by the ranks of the diagonal maps $V_\alpha \to V_{\alpha+(t,t)}$. Therefore, we can filter the Landscape by replacing the rank with the Skyscraper Invariant. As an example, we will use the IHC stained sample LargeHypoxicRegion1 of a tumour, for which the authors of [61] analyse the location of immune cells using Multiparameter Landscapes.

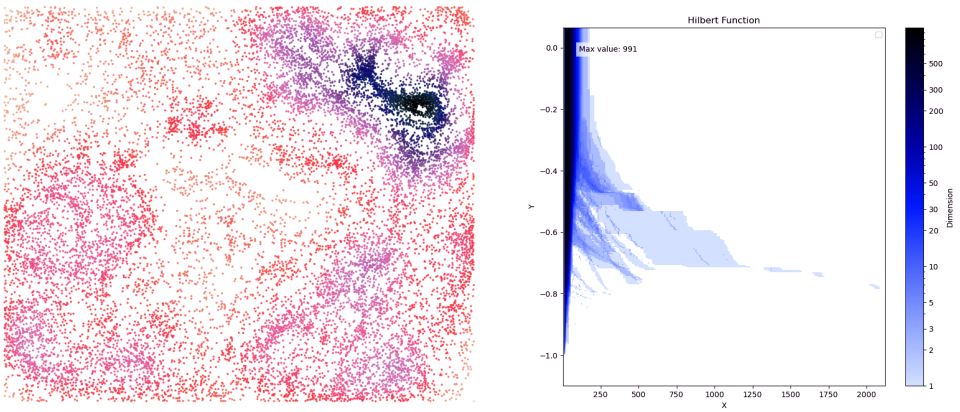


Figure 14 Immune cells in tumor tissue with density estimate and Hilbert function of $\mathbf{H}_1$.

Using MULTIPERS [46], we computed its density-alpha bifiltration with FUNCTION-DELAUNAY [63], a minimal presentation of the first persistent homology group thereof with MPFREE [40], and restricted it to a 300×300 grid. Computing the Skyscraper Invariant on this 35360×25553 matrix took around 5 minutes.

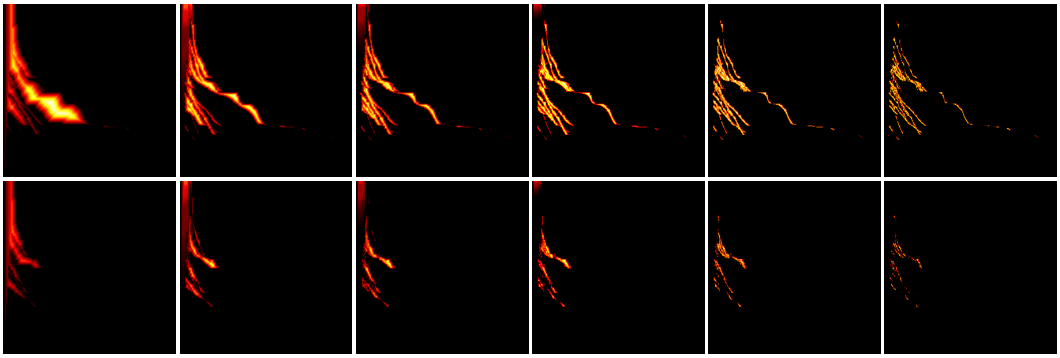


Figure 15 The filtered Landscape for $\mathbf{H}_1$ of Figure 14 with $\theta = 0, 450, 900, \ldots, 7200$ and $k = 1, 2$.

Since the filtered Landscapes are stable [26, Corollary 3.12], their differences are too, revealing elements of *long* lifetime.

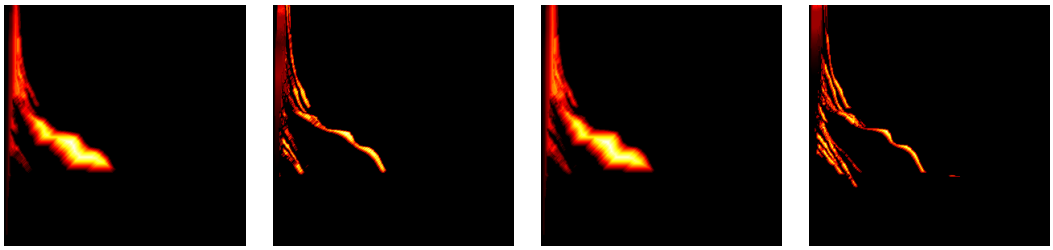


Figure 16 The differences $L_0 - L_1$, $L_1 - L_2$, $L_0 - L_2$, and $L_2 - L_4$ from the first row.

Conclusion. With our algorithms and the induced filtered Landscapes practitioners have a new invariant for topological data analysis that is robust to noise, finer than the rank invariant, efficiently computable, and is interpretable directly from its visual representation.

Open Problems. To incorporate invariants into Machine Learning pipelines, the authors of [58] define a framework which needs to be extended to include the Skyscraper Invariant. Instead of Landscapes, one can also filter other vectorizations of the rank invariant. By changing the stability condition [26], we obtain different HN filtrations and invariants. Can our methods be adapted to this general setting?

For each $\theta \in \mathbb{R}$, the Skyscraper invariant can be Möbius inverted [57], just like the rank invariant [48]. What are the properties of this path?

References

1 Ethan André, Jingyi Li, David Loiseaux, and Steve Oudot. Estimating the persistent homology of $\mathbb{R}^n$-valued functions using function-geometric multifiltrations, 2025. `arXiv:2412.04162`.

2 Hideto Asashiba. Relative koszul coresolutions and relative betti numbers. *Journal of Pure and Applied Algebra*, 229(3):107905, 2025. `doi:10.1016/j.jpaa.2025.107905`.

3 Ulrich Bauer, Fabian Lenzen, and Michael Lesnick. Efficient Two-Parameter Persistence Computation via Cohomology. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:17, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2023.15.

4 Ulrich Bauer and Luis Scoccola. Multi-parameter persistence modules are generically indecomposable. *International Mathematics Research Notices*, 2025(5):rnaf034, February 2025. doi:10.1093/imrn/rnaf034.

5 Matías Bender, Oliver Gäfvert, and Michael Lesnick. Computing minimal presentations of multi-parameter persistent homology, 2023. URL: https://github.com/MBender/muphasa.

6 Katherine Benjamin, Aneesha Bhandari, Jessica D. Kepple, Rui Qi, Zhouchun Shang, Yanan Xing, Yanru An, Nannan Zhang, Yong Hou, Tanya L. Crockford, Oliver McCallion, Fadi Issa, Joanna Hester, Ulrike Tillmann, Heather A. Harrington, and Katherine R. Bull. Multiscale topology classifies cells in subcellular spatial transcriptomics. *Nature*, 630(8018):943–949, June 2024. doi:10.1038/s41586-024-07563-1.

7 Håvard Bakke Bjerkevik. Stabilizing decomposition of multiparameter persistence modules. *Foundations of Computational Mathematics*, January 2025. doi:10.1007/s10208-025-09695-w.

8 Benjamin Blanchette, Thomas Brüstle, and Eric J. Hanson. Exact structures for persistence modules. In Lidia Angeleri Hügel, Bernhard Keller, María Julia Redondo, and Andrea Solotar, editors, *Representations of Algebras and Related Topics*, pages 121–160. European Mathematical Society Publishing House, 2022. doi:10.4171/ECR/21/4.

9 Magnus Bakke Botnan, Steffen Oppermann, and Steve Oudot. Signed Barcodes for Multi-Parameter Persistence via Rank Decompositions. In *International Symposium on Computational Geometry (SoCG)*, 2022. doi:10.4230/LIPIcs.SoCG.2022.19.

10 Tom Bridgeland. Stability conditions on triangulated categories. *Annals of Mathematics*, 166:317–345, 2007. doi:10.4007/annals.2007.166.317.

11 Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. *Discrete and Computational Geometry*, 42:71–93, June 2007. doi:10.1007/s00454-009-9176-0.

12 Gunnar E. Carlsson and Afra Zomorodian. The theory of multidimensional persistence. In *The ACM Symposium on Computational Geometry (SoCG)*, 2007. doi:10.1145/1247069.1247105.

13 Mathieu Carrière and Andrew J. Blumberg. Multiparameter persistence images for topological machine learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc., 2020. URL: https://proceedings.neurips.cc/paper_files/paper/2020/hash/fdff71fcab656abfbefaabecab1a7f6d-Abstract.html.

14 Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J Guibas, and Steve Y Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 237–246, 2009. doi:10.1145/1542362.1542407.

15 Chi-Yu Cheng. A deterministic algorithm for Harder-Narasimhan filtrations for representations of acyclic quivers. *Algebra & Number Theory*, 18(2):319–347, 2024. doi:10.2140/ant.2024.18.319.

16 René Corbet, Ulderico Fugacci, Michael Kerber, Claudia Landi, and Bei Wang. A kernel for multi-parameter persistent homology. *Computers & Graphics: X*, 2, 2019. doi:10.1016/j.cagx.2019.100005.

17 René Corbet, Michael Kerber, Michael Lesnick, and Georg Osang. Computing the multicover bifiltration. *Discrete & Computational Geometry*, 70(2):376–405, September 2023. doi:10.1007/s00454-022-00476-8.

18 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, 3 edition, 2008.

19 Harm Derksen and Visu Makam. Polynomial degree bounds for matrix semi-invariants. *Advances in Mathematics*, 310:44–63, 2017. doi:10.1016/j.aim.2017.01.018.

20 Tamal K. Dey, Jan Jendrysiak, and Michael Kerber. Decomposing Multiparameter Persistence Modules. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 41:1–41:19, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2025.41.

21 Tamal K. Dey, Woojin Kim, and Facundo Mémoli. Computing generalized rank invariant for 2-parameter persistence modules via zigzag persistence and its applications. In *International Symposium on Computational Geometry (SoCG)*, 2022. doi:10.4230/LIPIcs.SOCG.2022.34.

22 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, pages 454–463. IEEE, 2000. doi:10.1007/s00454-002-2885-2.

23 Burçin Eröcal, Oleksandr Motsak, Frank-Olaf Schreyer, and Andreas Steenpaß. Refined algorithms to compute syzygies. *Journal of Symbolic Computation*, 74:308–327, 2016. doi:10.1016/j.jsc.2015.07.004.

24 Gerd Faltings. Mumford-stabilität in der algebraischen geometrie. In S. D. Chatterji, editor, *Proceedings of the International Congress of Mathematicians*, pages 648–655, Basel, 1995. Birkhäuser Basel.

25 Marc Fersztand. Cheng's algorithm for persistence modules over finite posets. Software, version 0.1., EPSRC grant EP/R018472/1., swhId: swh:1:dir:b620e11fefcb30adcafaf299484f00c1b0a2af70 (visited on 2026-05-18). URL: https://github.com/marcf-ox/sky-inv-quiv, doi:10.4230/artifacts.26093.

26 Marc Fersztand. Harder-narasimhan filtrations of persistence modules: metric stability, 2024. arXiv:2406.05069.

27 Marc Fersztand, Emile Jacquard, Vidit Nanda, and Ulrike Tillmann. Harder–Narasimhan filtrations of persistence modules. *Transactions of the London Mathematical Society*, 11(1):e70003, 2024. doi:10.1112/tlm3.70003.

28 Marc Fersztand and Jan Jendrysiak. Computing the skyscraper invariant, 2026. arXiv:2603.23560.

29 Cole Franks, Tasuku Soma, and Michel X Goemans. Shrunk subspaces via operator sinkhorn iteration, 2022. doi:10.48550/arXiv.2207.08311.

30 Cole Franks, Tasuku Soma, and Michel X Goemans. Shrunk subspaces via operator sinkhorn iteration. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1655–1668. SIAM, 2023. doi:10.1137/1.9781611977554.ch62.

31 Günter Harder and Mudumbai S. Narasimhan. On the cohomology groups of moduli spaces of vector bundles on curves. *Mathematische Annalen*, 212:215–248, 1974. doi:10.1007/BF01357141.

32 Lutz Hille and José Antonio de la Peña. Stable representations of quivers. *Journal of Pure and Applied Algebra*, 172(2):205–224, 2002. doi:10.1016/S0022-4049(01)00167-0.

33 Alana Huszar. Non-commutative rank and semi-stability of quiver representations, 2021. arXiv:2111.00039.

34 Gábor Ivanyos, Youming Qiao, and KV Subrahmanyam. Constructive non-commutative rank computation is in deterministic polynomial time. *computational complexity*, 27:561–593, 2018. doi:10.1007/s00037-018-0165-7.

35 Yuni Iwamasa, Taihei Oki, and Tasuku Soma. Algorithmic Aspects of Semistability of Quiver Representations. In *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*, volume 334 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 99:1–99:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.99.

36 Yuni Iwamasa, Taihei Oki, and Tasuku Soma. Algorithmic aspects of semistability of quiver representations, 2025. arXiv:2407.06493.

37 Emile Jacquard. *Topics in multidimensional persistence*. Phd thesis, University of Oxford, 2024. URL: https://ora.ox.ac.uk/objects/uuid:591b13e1-faab-4da2-a405-d986a629003a.

38 Jan Jendrysiak. Skyscraper Invariant and Filtered Multiparameter Landscapes. Software, version 0.1., swhId: swh:1:dir:8ed2562604075467f0b0014975100a2d26e434ba (visited on 2026-05-18). URL: https://github.com/JanJend/Skyscraper-Invariant, doi:10.4230/artifacts.26094.

39 Michael Kerber and Tung Lam. Benchmark data sets of minimal presentations of 2-parameter persistence modules [data set]. Graz University of Technology, 2025. doi:10.3217/rxedk-qyq77.

40 Michael Kerber and Alexander Rolle. Fast minimal presentations of bi-graded persistence modules. In *Algorithm Engineering and Experiments (ALENEX)*, 2021. doi:10.1137/1.9781611976472.16.

41 Woojin Kim and Facundo Mémoli. Generalized persistence diagrams for persistence modules over posets. *Journal of Applied and Computational Topology*, 5(4):533–581, August 2021. doi:10.1007/s41468-021-00075-1.

42 A. D. King. Moduli of representations of finite dimensional algebras. *The Quarterly Journal of Mathematics*, 45(4):515–530, December 1994. doi:10.1093/qmath/45.4.515.

43 Michael Lesnick and Matthew Wright. Computing minimal presentations and bigraded betti numbers of 2-parameter persistent homology. *SIAM Journal on Applied Algebra and Geometry*, 6(2):267–298, 2022. doi:10.1137/20M1388425.

44 Michael Lesnick and Matthew Wright. Fast queries of fibered barcodes, 2025. arXiv:2511.05837.

45 David Loiseaux, Mathieu Carrière, and Andrew J Blumberg. Multi-parameter module approximation: an efficient and interpretable invariant for multi-parameter persistence modules with guarantees. *Journal of Applied and Computational Topology*, 9(4):1–60, 2025. doi:10.1007/s41468-025-00222-y.

46 David Loiseaux and Hannah Schreiber. Multipers: Multiparameter Persistence for Machine Learning. *Journal of Open Source Software*, 9(103):6773, November 2024. doi:10.21105/joss.06773.

47 David Loiseaux, Luis Scoccola, Mathieu Carrière, Magnus Bakke Botnan, and Steve Oudot. Stable vectorization of multiparameter persistent homology using signed barcodes as measures. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/d75c474bc01735929a1fab5d0de3b189-Abstract-Conference.html.

48 Alexander McCleary and Amit Patel. Edit distance and persistence diagrams over lattices. *SIAM Journal on Applied Algebra and Geometry*, 6(2):134–155, 2022. doi:10.1137/20M1373700.

49 Ezra Miller. The alexander duality functors and local duality with monomial support. *Journal of Algebra*, 231(1):180–234, 2000. doi:10.1006/jabr.2000.8359.

50 Ezra Miller. Graded greenlees–may duality and the čech hull. In *Local Cohomology and Its Applications*, volume 226 of *Lecture Notes in Pure and Applied Mathematics*, pages 233–253. Marcel Dekker, 2001. doi:10.1201/9781482275766.

51 Ezra Miller. Data structures for real multiparameter persistence modules, 2020. arXiv:1709.08155.

52 Ezra Miller and Jiaxi Zhang. Lifetime filtration of multiparameter persistence modules, 2024. Presentation for Spires 2024, Oxford. URL: https://sites.math.duke.edu/~ezra/Talks/horizons-in-alg-for-multiPH.pdf.

53 Dmitriy Morozov and Amit Patel. Output-sensitive computation of generalized persistence diagrams for 2-filtrations, 2023. arXiv:2112.03980.

54 David Mumford, John Fogarty, and Frances Kirwan. *Geometric Invariant Theory*, volume 34 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer-Verlag, 3 edition, 1994.

55 Amit Patel. Generalized persistence diagrams. *Journal of Applied and Computational Topology*, 1(3):397–419, 2018. doi:10.1007/s41468-018-0012-6.

56 Ville Puuska. Erosion distance for generalized persistence modules. *Homology, Homotopy & Applications*, 22(1), 2020. doi:10.4310/HHA.2020.v22.n1.a14.

57 Gian Carlo Rota. On the foundations of combinatorial theory i. theory of möbius functions. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 2(4):340–368, January 1964. doi:10.1007/BF00531932.

58 Luis Scoccola, Siddharth Setlur, David Loiseaux, Mathieu Carrière, and Steve Oudot. Differentiability and optimization of multiparameter persistent homology. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024. URL: https://proceedings.mlr.press/v235/scoccola24a.html.

59 Donald R. Sheehy. A multicover nerve for geometric inference. In *CCCG: Canadian Conference in Computational Geometry*, pages 309–314, 2012. URL: http://2012.cccg.ca/papers/paper52.pdf.

60 Oliver Vipond. Multiparameter persistence landscapes. *Journal of Machine Learning Research*, 21:1–38, 2020. URL: http://jmlr.org/papers/v21/19-054.html.

61 Oliver Vipond, Joshua A Bull, Philip S Macklin, Ulrike Tillmann, Christopher W Pugh, Helen M Byrne, and Heather A Harrington. Multiparameter persistent homology landscapes identify immune cell spatial patterns in tumors. *Proceedings of the National Academy of Sciences*, 118(41), October 2021. doi:10.1073/pnas.2102166118.

62 Cheng Xin, Soham Mukherjee, Shreyas N. Samaga, and Tamal K. Dey. GRIL: A 2-parameter persistence based vectorization for machine learning. In *Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*, pages 313–333, 2023. URL: https://proceedings.mlr.press/v221/xin23a.html.

63 Ángel Javier Alonso, Michael Kerber, Tung Lam, and Michael Lesnick. Delaunay bifiltrations of functions on point clouds. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4872–4891, 2024. doi:10.1137/1.9781611977912.173.

FPT Approximations for Capacitated Sum of Radii and Diameters

Arnold Filtser
Bar-Ilan University, Ramat Gan, Israel

Ameet Gadekar
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Abstract

The Capacitated Sum of Radii problem involves partitioning a set of points P, where each point $p \in P$ has capacity U_p, into k clusters that minimize the sum of cluster radii, such that the number of points in the cluster centered at point p is at most U_p. We begin by showing that the problem is APX-hard, and that under gap-ETH there is no parameterized approximation scheme (FPT-AS). We then construct a ≈ 5.83-approximation algorithm in FPT time (improving a previous ≈ 7.61 approximation in FPT time). Our results also hold when the objective is a general monotone symmetric norm of radii. We also improve the approximation factors for the uniform capacity case, and for the closely related problem of Capacitated Sum of Diameters.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Approximation algorithms analysis; Theory of computation → Fixed parameter tractability; Theory of computation → Facility location and clustering

Keywords and phrases clustering, sum of radii, sum of diameter, capacitated clustering, fpt

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.48

Related Version *The reader is encouraged to read the full version of the paper*: `https://arxiv.org/pdf/2409.04984`

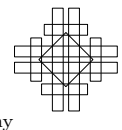
Funding *Arnold Filtser*: This research was supported by the Israel Science Foundation (Grant No. 1042/22).
Ameet Gadekar: This work was partially supported by the Israel Science Foundation (Grant No. 1042/22) while the author was affiliated with Bar-Ilan University.

Acknowledgements The authors thank the anonymous reviewer for suggesting the construction used in Theorem 2, which improves upon the one presented in an earlier version of the paper.

1 Introduction

Clustering is a fundamental problem in several domains of computer science, including, data mining, operation research, and computational geometry, among others. In particular, center based clustering problems such as k-median, k-means, and k-center have received significant attention from the research community for more than half a century [66, 55, 69, 9, 7, 61, 38, 53, 68, 59, 23, 15, 33, 32, 31, 34, 25, 1, 49]. In these problems, we are given a set P of n points together with a distance function (metric) and a positive integer k. The goal is to partition P into k parts called *clusters* and choose a *center* point for each cluster, so that to minimize a clustering objective that is a function of the point distances to their centers. A related fundamental problem that helps reduce *dissection effect* due to k-center [70, 28] (see Figure 1) is called Sum of Radii. Here the goal is to choose k-size subset X of P (called centers, as before) and assign every point to an element in X. This partitions the set P into possible k clusters, $C_1, \cdots, C_k$, where cluster C_i corresponds to the set of points assigned to $x_i \in X$. The radius of cluster C_i centered at $x_i \in X$ is the maximum distance of a point in


© Arnold Filtser and Ameet Gadekar;
licensed under Creative Commons License CC-BY 4.0

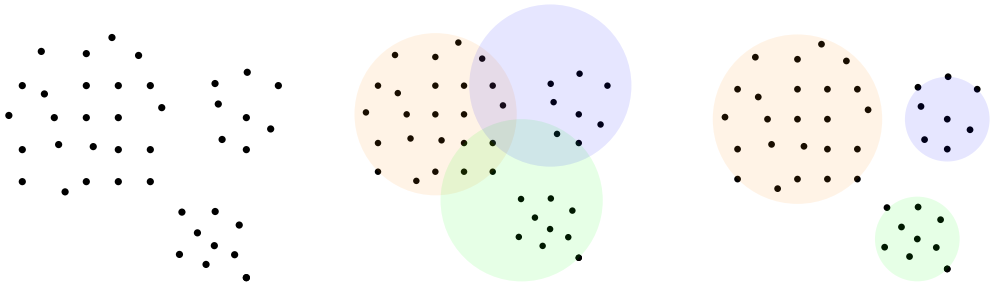


Figure 1 The optimal 3-center solution (middle) splits the ground-truth clusters, whereas the Sum of Radii objective (right) avoids this dissection.

P_i to x_i. The objective is the sum of radii of the clusters $C_1, \cdots, C_k$.[1]

In the recent years Sum of Radii received a great share of interest in all aspects [28, 45, 46, 46, 14, 43, 21, 29, 56, 11, 58, 12, 3, 17, 42, 18]. Nevertheless, its computational landscape is not yet fully understood. While, the problem is NP-hard [45] even in weighted planar graphs and in metrics of bounded doubling dimensions, it is known to admit a QPTAS (quasi polynomial time approximation scheme) [45] in general metrics, thus prompting a possibility of PTAS (polynomial time approximation scheme).[2] This is in contrast to related clustering problems like k-center, k-median, and k-means which are all known to be APX-hard [55, 57, 8]. Currently, the present best known approximation factor in polynomial time is $(3+\varepsilon)$ due to Buchem, Ettmayr, Rosado, and Wiese [21] (improving over previous results of [28, 43]). Additionally, there is a recent $(2+\varepsilon)$-approximation algorithm [29] that runs in FPT time[3] (fixed parameter tractable).

Capacitated clustering

We are interested in a much more challenging generalization where each point p has an inherent capacity $U_p \geq 0$, indicating an upper bound on the number of points it can serve as a center of a cluster in the solution.[4] This is known as the *Non-Uniform Capacitated* clustering problem. If all the points have the same capacity U, the problem is referred to as the *Uniform Capacitated* clustering problem. Capacitated clustering naturally models many applications: in *load balancing*, cluster centers correspond to servers with limited service capacity, the radius captures latency, and minimizing the sum of radii models reducing aggregate latency subject to serve load; in *wireless sensors*, each cluster head can accommodate only bounded number of devices due to bandwidth, memory, or battery limitations, the radius corresponds to communication range, and optimizing the sum of radii reflects minimizing total communication energy. Capacitated clustering is thoroughly studied: Capacitated k-center admits a constant factor approximation in polynomial time for both uniform [60] and non-uniform capacities [37, 5]. There are several bi-criteria polytime approximations for k-median [30, 22, 24, 39, 63, 64].

---

[1] Alternatively, in the Sum of Radii problem we choose centers $x_1, \ldots, x_k \in X$, and radii $r_1, \ldots, r_k \in \mathbb{R}_{\geq 0}$ so that $P \subseteq \cup_{i=1}^{k} \mathsf{ball}(x_i, r_i)$. The objective is to minimize $\sum_{i=1}^{k} r_i$.

[2] If Sum of Radii is APX-hard (equivalently, does not admit a PTAS) then NP$\subseteq$QP (in particular there is a quasi-polynomial time algorithm solving SAT). It is widely believed that NP$\not\subseteq$QP.

[3] In this paper, by FPT time, we mean FPT w.r.t. the parameter k.

[4] Formally, following Footnote 1, a valid solution to the capacitated version also contains an assignment $\sigma : P \to \{x_1, \ldots, x_k\}$ such that if $\sigma(p) = x_i$ then $\delta(p, x_i) \leq r_i$, and for every i, $|\sigma^{-1}(x_i)| \leq U_{x_i}$.

There are FPT approximation algorithms for capacitated k-median and capacitated k-means with approximation factors $(3+\varepsilon)$ and $(9+\varepsilon)$ respectively [35] (improving over [2, 73]). On the other hand, assuming gap-ETH[5], in FPT time it is impossible to approximate k-median and k-means with factors better than $(1+\frac{2}{e})$ and $(1+\frac{8}{e})$, respectively (even without capacities) [32]. There is a $4+\varepsilon$ FPT-approximation for uniform capacitated k-center [58, 50][6].

For capacitated Sum of Radii (CapSoR) (see Table 1 for a summary of previous and new results), Inamdar and Varadarajan [56] constructed an FPT time algorithm providing an 28-approximation for Sum of Radii with uniform capacities. This was later improved to $4+\varepsilon$ by Bandyapadhyay, Lochet, and Saurabh [11], and finally to $3+\varepsilon$ by Jaiswal, Kumar, and Yadav [58]. For non-uniform capacities, the latter paper obtained $(4+\sqrt{13}+\varepsilon) \approx 7.61$ approximation in FPT time, improving over the previous approximation of $15+\varepsilon$ [11].

From the lower bound side, the authors in [58] showed that assuming ETH[7], there is some constant $\beta > 0$, such that any β-approximation for CapSoR (with non-uniform capacities) requires $2^{\Omega(k/\operatorname{poly}\log k)}n^{O(1)}$ time. The first contribution of this paper is to show that CapSoR is APX hard for an explicit factor and even with uniform capacities. That is compared to [58] we removed the ETH assumption, used only uniform capacities, and showed an explicit factor of inapproximability. This is in contrast to the uncapacitated version which is believed to admit a PTAS (assuming $NP \not\subseteq QP$).

▶ **Theorem 1** (APX-hard). *For every constant $\varepsilon > 0$, it is NP-hard to approximate Uniform CapSoR to within a factor of $(1+1/e-\varepsilon)$.*

Earlier, Bandyapadhyay, Lochet, and Saurabh [11] showed that assuming ETH[7], no FPT algorithm can solve CapSoR exactly (even with uniform capacities). However, until this point, nothing ruled out the existence of an FPT-Approximation Scheme (FPT-AS) or PAS for Parameterized Approximation Scheme[8]. The second contribution of this paper is to show that assuming gap-ETH[5], no such FPT-AS exists.

▶ **Theorem 2** (No FPT-AS). *Assuming gap-ETH, for every $\varepsilon > 0$ and any function f, there is no $f(k)n^{o(k)}$ time algorithm that approximates Uniform CapSoR to within a factor of $(1+1/e-\varepsilon)$.*

Note that if we allow $n^{O(k)}$ time, then we can exactly solve Sum of Radii using brute-force, even with non-uniform capacities. Therefore, Theorem 2 not only rules out FPT-approximation algorithms for achieving approximation better than factor of $(1+1/e-\varepsilon)$ for this problem, but also implies that the best algorithm in this context essentially is the naive brute-force algorithm, which runs in $n^{O(k)}$ time.

Next, as the best we can hope for is a constant factor approximation in FPT time, we turn to improving this factor. The main result of the paper is a $(3+\sqrt{8}+\varepsilon) \approx 5.83$ FPT-approximation for non-uniform CapSoR, significantly improving the present best [58]

[5] Informally, gap-ETH says that there exists an $\varepsilon > 0$ such that no sub-exponential time algorithm for 3-SAT can distinguish whether a given 3-SAT formula has a satisfying assignment or every assignment satisfies at most $(1-\varepsilon)$ fraction of the clauses.

[6] In fact, [50] obtained a 2-approximation for capacitated k-center under *soft* assignments (where different centers can be co-located). In contrast, in this paper we consider only *hard* assignments (where we can open only a single center at each point). One can use [50] to obtain a 4-approximation for uniform capacitated k-center w.r.t. hard assignments. See Section 3 for a further discussion.

[7] Informally, ETH (Exponential Time Hypothesis), says that there is no subexponential time algorithm for 3-SAT.

[8] Such algorithms find $(1+\varepsilon)$-approximation in time $f(k,\varepsilon)n^{g(\varepsilon)}$ for some fixed functions f and g.

factor of $(4+\sqrt{13}+\varepsilon) \approx 7.61$. The theorem also mentions cluster capacities and other objectives. These will be explained in the following sub-sections.

▶ **Theorem 3** (Main Theorem). *There is a (deterministic) FPT-approximation algorithm that finds a $(3+2\sqrt{2}+\varepsilon)$-approximation for Non-uniform CapSoR for any $\varepsilon > 0$, and runs in time $2^{O(k^2 \log k + k \log(k/\varepsilon))} n^{O(1)}$. Furthermore, the algorithm yields $(3+2\sqrt{2}+\varepsilon)$-approximation for non-uniform capacities even when the objective is a monotone symmetric norm of the radii.*
Both results hold also w.r.t. cluster capacities.

Table 1 Summary of our results and previous results for CapSoR and CapSoD. The results for CapSoD are implicit in the corresponding references. Furthermore, CapSoD makes sense only for the cluster capacities. While CapSoR with the ℓ_∞ objective corresponds to the Capacitated k-Center problem, CapSoD with the ℓ_∞ objective corresponds to the Capacitated Max-k-Diameter problem. While our algorithms are deterministic for non-uniform capacities, the algorithms for the uniform capacity case are randomized. The inapproximability result, assuming gap-ETH, rules out any $f(k)n^{o(k)}$ time algorithm that can approximate uniform CapSoR (even with ℓ_1 objective) to a factor better than $(1+1/e)$.

Capacities	Norm	Approx. factor	Run time	Ref.
Sum of Radii				
Uniform	ℓ_1	28	$2^{O(k^2)} n^{O(1)}$	[56]
		$4+\varepsilon$	$2^{O(k \log(k/\varepsilon))} n^{O(1)}$	[11]
		$3+\varepsilon$		
	ℓ_p	$(1+\varepsilon)(2 \cdot 4^{p-1}+1)^{1/p}$	$2^{O(k \log(k/\varepsilon))} n^{O(1)}$	[58]
		$(1+\varepsilon)(2 \cdot 3^{p-1}+1)^{1/p}$	$2^{O(k^2 + k \log(k/\varepsilon))} n^{O(1)}$	Theorem 4
	general	$3+\varepsilon$	$2^{O(k^2 + k \log(k/\varepsilon))} n^{O(1)}$	Theorem 4
(Non-Uni-) Hardness	general	$\beta > 1$	$2^{\Omega(k/\text{polylog}k)} \cdot \text{poly}(n)$ (ETH)	[58]
Hardness Uniform	general	exact	$n^{\Omega(k)}$ (ETH)	[11]
		$(1+1/e-\varepsilon)$	NP-hard	Theorem 1
		$(1+1/e-\varepsilon)$	$n^{\Omega(k)}$ (gap-ETH)	Theorem 2
Non-Uniform	ℓ_1	$15+\varepsilon$	$2^{O(k^2 \log(k/\varepsilon))} n^{O(1)}$	[11]
	ℓ_p	$4+\sqrt{13}+\varepsilon \approx 7.61$	$2^{O(k^3 + k \log(k/\varepsilon))} n^{O(1)}$	[58]
	general	$3+\sqrt{8}+\varepsilon \approx 5.83$	$2^{O(k^2 \log k + k \log(k/\varepsilon))} n^{O(1)}$	Theorem 3
Sum of Diameters				
Uniform	ℓ_1	$6+\varepsilon$	$2^{O(k \log(k/\varepsilon))} n^{O(1)}$	[58]
		$4+\varepsilon$	$2^{O(k \log(k/\varepsilon))} n^{O(1)}$	Theorem 5
	ℓ_p	$(2+\varepsilon)(2 \cdot 4^{p-1}+1)^{1/p}$	$2^{O(k \log(k/\varepsilon))} n^{O(1)}$	[58]
		$(2+\varepsilon)(2^{p-1}+1)^{1/p}$	$2^{O(k \log(k/\varepsilon))} n^{O(1)}$	Theorem 5
	general	$4+\varepsilon$	$2^{O(k \log(k/\varepsilon))} n^{O(1)}$	Theorem 5
Non-Uniform	general	$7+\varepsilon$	$2^{O(k^2 \log k + k \log(k/\varepsilon))} n^{O(1)}$	Theorem 6

1.1 Other norm objectives

In the Sum of Radii problem the goal is to choose k balls covering all the metric points such that the sum of radii $\sum_i r_i$ is minimized. A natural generalization is to optimize alternate objectives. Specifically, given a norm $\|\cdot\| : \mathbb{R}^k \to \mathbb{R}_{\geq 0}$, the task is to choose balls covering all metric points so as to minimize the norm of the radii vector $\|(r_1, \ldots, r_k)\|$. A canonical example is the ℓ_p norm where $\|(r_1, \ldots, r_k)\|_p = \left(\sum_{i=1}^k r_i^p\right)^{1/p}$, which has been studied over two decades [19, 13]. The ℓ_1 norm objective recovers the Sum of Radii problem, while the ℓ_∞ norm objective ($\|(r_1, \ldots, r_k)\|_\infty = \max_{i\in[1,k]} r_i$) corresponds to the classical k-center problem. The other ℓ_p-norm objectives interpolate between these two fundamental problems in clustering. In particular, they capture objectives such as sum of squared radii,[9] which has received attention in practice, especially in wireless network applications where it naturally models power consumption [4, 65]. This perspective further extends to objectives that impose non-uniform weights on the radii, such as ordered weighted norms or cascaded multi-level ℓ_p norms. These allow the norm objective to encode heterogeneous priorities or costs across clusters in practical settings [54]. The strength of such generalizations has been recently studied [27, 1], which not only unified the existing FPT-AS for several different problems such as k-median, k-means, and k-center, but also lead to FPT-AS for advanced problems such as priority k-center [48, 10, 72], ℓ-centrum [71, 62], ordered k-median [25, 20], and Socially Fair k-median [6, 16, 67, 44], many of which were previously unresolved.

The authors in [58] constructed an algorithm for the uniform capacitated Sum of Radii with ℓ_p norm of radii with approximation factor $(1+\varepsilon)(2\cdot 4^{p-1}+1)^{1/p}$ in FPT time. In particular, this implies a $(3+\varepsilon)$ FPT-approximation for uniform capacitated Sum of Radii ($p=1$) and $(4+\varepsilon)$ FPT-approximation[6] for uniform capacitated k-center ($p=\infty$). In our work, we generalize and improve these results. Specifically, we obtain a $(3+\varepsilon)$ FPT-approximation w.r.t. any monotone symmetric norm of radii, thus generalizing the ℓ_p norm objective of [58]. Further, we also improve the approximation ratio for the ℓ_p norm objective to $(1+\varepsilon)(2\cdot 3^{p-1}+1)^{1/p}$.

▶ **Theorem 4** (Uniform). *There is a randomized algorithm for Uniform CapSoR, oblivious to the objective, that runs in $2^{O(k^2+k\log(k/\varepsilon))}n^{O(1)}$ time, and with probability at least $3/5$ returns a solution* Sol*, such that* Sol *is a $(3+\varepsilon)$-approximation w.r.t. any monotone symmetric norm objective (simultaneously). Furthermore, for $1<p<\infty$,* Sol *is a $(1+\varepsilon)(2\cdot 3^{p-1}+1)^{1/p}$-approximation w.r.t. the ℓ_p norm objective.*

For the special case of $p=\infty$, the ℓ_∞ norm objective is simply the k-center problem. Here a simple corollary of Theorem 4 implies $(3+\varepsilon)$ FPT-approximation algorithm for k-center with uniform capacities, improving the state of the art factor of 4 due to [58, 50]. Note that for $p\in(1,\infty]$, the approximation factor of our Theorem 4 is better than that of [58]. In fact, the approximation factor of our algorithm for ℓ_p norm objective equals $\max_{\alpha\in[0,1]}\left(\frac{(2+\alpha)^p+1}{1+\alpha^p}\right)^{1/p}$, which is slightly better than the stated factor. For example, it equals ≈ 2.414 and ≈ 2.488 for $p=2$ and $p=3$, respectively (instead of the stated ≈ 2.65 and ≈ 2.67). Similarly, [58] explicitly claimed approximation factors of 3 and ≈ 3.2, respectively. However, if we optimize their final expression, it yields factors ≈ 2.92 and ≈ 3.191 for $p=2$ and $p=3$, respectively. See the full version for further discussion.

[9] Sum of squared radii is essentially equivalent to the ℓ_2 objective: an α-approximation algorithm for the ℓ_2 objective yields an α^2-approximation to the sum of squared radii.

For non-uniform Capacitated Sum of Radii, [58] obtained approximation factor of $(4+\sqrt{13}+\varepsilon) \approx 7.61$ w.r.t. any ℓ_p-norm objective in $2^{O(k^3+k\log(k/\varepsilon))}n^{O(1)}$ time. As stated in Theorem 3 above, for non-uniform Capacitated Sum of Radii, we obtain approximation factor of $(3+\sqrt{8}+\varepsilon) \approx 5.83$ w.r.t. any monotone symmetric norm objective, in $2^{O(k^2\log k+k\log(k/\varepsilon))}n^{O(1)}$ time. Thus we improve over [58] on three fronts (see Table 1): (1) approximation factor, (2) generalizing to any monotone symmetric norm from ℓ_p norms, and (3) running time.

1.2 Sum of Diameters and cluster capacities

A closely related problem is the Sum of Diameters problem, which has been studied extensively and predates Sum of Radii [51, 52, 70, 26, 40, 28, 14, 40, 43]. Here, the goal is to partition the point set P into k clusters $C_1, \ldots, C_k$, and the objective is to minimize the sum of cluster diameters $\sum_{i=1}^{k} \mathrm{diam}(C_i)$, where the diameter of a cluster is $\mathrm{diam}(C_i) = \max_{x,y\in C_i} \delta(x,y)$ the maximum pairwise distance between two cluster points. Note that, unlike Sum of Radii, in this problem, there are no centers representing the clusters. Furthermore, this problem is NP-hard to approximate to a factor better than 2 in polynomial time [40] (unlike Sum of Radii, which admits a QPTAS). A simple observation shows that any α-factor approximation for Sum of Radii implies 2α-factor approximation for Sum of Diameters in a black-box way (and vice-versa).[10] This trick has often been used to design approximation algorithms for Sum of Diameters. In fact, the current state-of-the-art algorithms for Sum of Diameters, including a polynomial-time $(6+\varepsilon)$-approximation [21] and a quasi-polynomial-time $(2+\varepsilon)$-approximation [45], are based on this implicit trick by applying it to the polynomial-time $(3+\varepsilon)$-approximation and QPTAS for Sum of Radii, respectively.

Capacitated Sum of Diameters

We introduce the problem of Capacitated Sum of Diameters (CapSoD). Here we are initially given k capacities $U_1, \ldots, U_k$, the goal is to partition the point set P into k clusters $C_1, \ldots, C_k$, such that for every i, $|C_i| \le U_i$, while the objective is to minimize the sum of cluster diameters. We call the capacities uniform if all the capacities are equal $U_1 = \cdots = U_k$, and non-uniform otherwise. Under uniform capacities, the reduction mentioned in Footnote 10 goes through. Thus an α approximation algorithm to uniform capacitated Sum of Radii transfers in a black-box manner into a 2α approximation algorithm for the uniform capacitated Sum of Diameters, with the same running time. In particular, by using [58], one can obtain a $6+\varepsilon$-approximation for uniform capacitated Sum of Diameters in FPT time. In fact, similarly to Sum of Radii, one can study Capacitated Sum of Diameters w.r.t. any norm objective, and the reduction will still go through. Thus it follows from [58] that for any ℓ_p norm objective, uniform capacitated Sum of Diameters admits $2(1+\varepsilon)(4^{p-1}+1)^{1/p}$ FPT approximation. Similarly, using our Theorem 4, we can obtain FPT time $6+\varepsilon$ approximation for uniform capacitated Sum of Diameters w.r.t. to any monotone symmetric norm, or $2(1+\varepsilon)(3^{p-1}+1)^{1/p}$-approximation w.r.t. ℓ_p norm objective. Sum of Diameters is a fundamental and important problem. Its capacitated version was not previously explicitly studied simply because there was nothing to say beyond this simple reduction.

[10] Given a set of points P, denote by $\mathcal{R}$ and $\mathcal{D}$ the value of the optimal solutions to Sum of Radii and Sum of Diameters respectively. It holds that $\mathcal{R} \le \mathcal{D} \le 2\cdot\mathcal{R}$. Indeed, consider an optimal solution to Sum of Diameters of cost $\mathcal{D}$, then by picking arbitrary center in each cluster we obtain a solution for Sum of Radii of cost at most $\mathcal{D}$ (thus $\mathcal{R} \le \mathcal{D}$). On the other hand, given a solution to Sum of Radii of cost $\mathcal{R}$, the clusters induced by the balls constitute a solution to Sum of Diameters of cost at most $2\cdot\mathcal{R}$ (thus $\mathcal{D} \le 2\cdot\mathcal{R}$).

In our work, we go beyond this reduction and directly design novel approximation algorithms for capacitated Sum of Diameters with significantly better approximation factors than twice that of Sum of Radii.

▶ **Theorem 5** (Uniform Diameters). *There is a randomized algorithm that given an instance of the Uniform CapSoD runs in $2^{O(k\log(k/\varepsilon))}n^{O(1)}$ time, and with probability at least 3/5 returns a solution* Sol*, such that* Sol *is a $(4+\varepsilon)$-approximation w.r.t. any monotone symmetric norm objective (simultaneously). Furthermore, for $1 < p < \infty$,* Sol *is a $(2+\varepsilon)(2^{p-1}+1)^{1/p}$-approximation w.r.t. the ℓ_p norm objective.*

Finally, we proceed to consider the more challenging problem of non-uniform Capacitated Sum of Diameters. Here we obtain a $7+\varepsilon$ approximation w.r.t. to any monotone symmetric norm objective.

▶ **Theorem 6** (Non-Uniform Diameters). *For any $\varepsilon > 0$, there is a (deterministic) algorithm running $2^{O(k^2+k\log(k/\varepsilon))}n^{O(1)}$-time and returns a $(7+\varepsilon)$-approximation for Non-uniform CapSoD w.r.t. any monotone symmetric norm objective.*

1.2.0.1 Cluster capacities

Bandyapadhyay, Lochet, and Saurabh [11] introduced the problem of Capacitated Sum of Radii where each point $p \in P$ has a capacity U_p, and a cluster centered in p can contain at most U_p points. This corresponds for example to a scenario where we want to construct water wells, and a well constructed at point p can serve up to U_p clients. However, an equally natural problem is where one is given k capacities $U_1, \ldots, U_k$, and the goal is to construct k clusters with arbitrary centers, such that the i'th cluster contains at most U_i points. This problem is similar to our capacitated Sum of Diameters, and can correspond to a scenario where one wants to distribute already existing k water tanks (for example in a tent village during a festival). We refer to the two versions of the problem as node capacities, and cluster capacities, respectively. Note that for uniform capacities the two versions coincide. Further, note that the reduction from Sum of Radii to Sum of Diameters mentioned in Footnote 10 holds in the capacitated version w.r.t. cluster capacities. Our results on node capacities in Theorem 3 hold for cluster capacities as well ($3+2\sqrt{2}+\varepsilon$ approximation for any monotone symmetric norm objective). No results on Sum of Radii with non-uniform cluster capacities were previously known.

Organization. Due to space constraints, we defer all algorithms and technical proofs to the full version. In the next section, we outline the main technical ideas behind our results.

2 Overview of techniques

In this section, we highlight our conceptual and technical contributions. Due to space constraints, we begin with our main theorem (Theorem 3) and present the key ideas behind our algorithm for non-uniform capacities in Section 2.1. Then, in Section 2.2, we delve into the algorithmic ideas for uniform capacities. Note however, that from a pedagogical viewpoint, it is easier to begin reading first Section 2.2, followed by Section 2.1. We now set up basic notations required for the exposition.

Notations. We denote by $\mathcal{I}$ an instance of capacitated Sum of Radii or capacitated Sum of Diameters, depending upon the context. For uniform capacities, $\mathcal{I}$ consists of metric space (P,δ), a positive integer k, and a uniform capacity $U > 0$. The elements in P are called points. For non-uniform node capacities, U is replaced by corresponding node capacities

$\{U_p\}_{p\in P}$, and for non-uniform cluster capacities, U is replaced by corresponding cluster capacities $\{U_1, \cdots, U_k\}$. We denote by $\mathcal{C} = \{C_1, \cdots, C_k\}$ an optimal (but fixed) clustering for $\mathcal{I}$. For ease of analysis, we assume that $|\mathcal{C}| = k$, otherwise, we can add zero radius clusters to make it k, without increasing the cost of $\mathcal{I}$. Note that the clusters in $\mathcal{C}$ are disjoint. When $\mathcal{I}$ is an instance of capacitated Sum of Radii, we denote by r_i^* and o_i as the radius and the center of cluster $C_i \in \mathcal{C}$, respectively, and let $O = \{o_1, \cdots, o_k\}$. In this case, we let $\text{OPT} = r_1^* + \cdots + r_k^*$. When $\mathcal{I}$ is an instance of capacitated Sum of Diameters, we denote by d_i^* as the diameter of cluster $C_i \in \mathcal{C}$. In this case, we let $\text{OPT} = d_1^* + \cdots + d_k^*$. It is known that we can guess in FPT time in k, a set (ε-approximation) $\{r_1, \cdots, r_k\}$ corresponding to $\{r_1^*, \cdots, r_k^*\}$ such that $\sum_{i\in[k]} r_i \leq (1+\varepsilon) \cdot \sum_{i\in[k]} r_i^*$ and $r_i \geq r_i^*$. Similarly, let $\{d_1, \cdots, d_k\}$ denote ε-approximation of $\{d_1^*, \cdots, d_k^*\}$. A feasible solution to $\mathcal{I}$ is a partition of P (along with centers for Sum of Radii) such that the capacities are respected. For a point $p \in P$ and a positive real r, denote by $\mathsf{ball}(p, r)$ as the set of points from P that are at a distance at most r from p.

▶ Remark 7 (Point assignment using matching and flows). In this section (and throughout the paper), we will focus on presenting a set of centers and their corresponding radii as our solution. This approach suffices because, given such a set of centers and radii, we can determine in polynomial time whether there exists a corresponding feasible solution (i.e., feasible assignment of points to the centers/clusters) with the same cost. Moreover, we can also find such a solution by defining an appropriate flow problem (see the full version).

▶ Remark 8 (FPT and bounded guessing). We assume that our algorithms have a power to make $\text{poly}(k)$ guesses each with success probability $p(k)$. Such algorithms can be transformed into a randomized FPT algorithms that are correct with constant probability. See the full version for more details.

2.1 Non-uniform capacitated Sum of Radii

For ease of exposition, in this technical overview we highlight our ideas for cluster capacities (see paragraph 1.2.0.1). Transitioning to node capacities introduces several more challenges which we will not discuss here. Other than the basic FPT framework mentioned in the preliminaries above, our algorithm is fundamentally different from these of [11, 58], and hence we do not attempt to compare them (in contrast, for uniform capacities the algorithms are similar).

Consider the following simple and natural strategy of processing clusters in $\mathcal{C}$ iteratively (see the full version for pseudo-code). Initially all the clusters in $\mathcal{C}$ are unprocessed (colored red). We will process the clusters in $\mathcal{C}$ in non-decreasing order of their radii, denoted as $C_1, \cdots, C_k$. Let $U_1, \cdots, U_k$ be the corresponding cluster capacities. Consider the first iteration when we process C_1. Suppose we could find a dense ball $D_1 = \mathsf{ball}(y_1, r_1)$ of radius r_1 in P such that $|D_1| \geq |C_1|$. Then, we could create a cluster $C_1' \subseteq D_1$ of size $|C_1|$[11] with radius r_1, matching the optimal cost. To make this cluster permanent, we delete the points from D_1 to prevent them from being reassigned in later iterations. However, this could create problems – (i) C_1' may contain points from other clusters, so in the future iterations when we process these clusters then the densest ball may not have enough points, and (ii) we need to make sure the points of C_1 are taken care by some cluster in our solution. Our algorithm is based on the following two key ideas that handle these two issues:

[11] Even though $|C_1|$ is unknown, in retrospect, we will be able to use Remark 7 to obtain such a clustering.

Invariant: do not touch the unprocessed. We ensure that, throughout the algorithm, every unprocessed cluster in $\mathcal{C}$ has all of its points intact. In other words, during the processing iteration of C_i, all its points are present.

Making progress: Good dense balls. An immediate implication of the above invariant is that when we process C_i, there is a ball of radius r_i containing at least $|C_i|$ points. Let $D_i = \mathsf{ball}(y_i, r_i)$ be the densest ball w.r.t. the remaining points during the iteration of C_i. Note that such a ball contains at least $|C_i|$ points due to the above invariant. Consider the smallest radius cluster $C_j \in \mathcal{C}$ that intersects D_i, and call C_j the *anchor* for D_i. See top-left figure in Figure 3, where D_i is the ball centered at y with radius r_i, $C_i = C_5$ and $C_j = C_8$. Now, consider the extended ball $D'_i := \mathsf{ball}(y_i, r_i + 2r_j)$, and note that $C_j \subseteq D'_i$. Now, if C_i intersects D'_i, then C_i is not far from y_i (see top-right figure in Figure 3). In this case, we call D_i a good dense ball, and will be able to process C_i. Thus, our first goal when processing the cluster C_i is to find a good dense ball.

Finding good dense ball in FPT time. Suppose D'_i does not intersect C_i, then we can temporarily delete $D'_i = \mathsf{ball}(y_i, r_i + 2r_j)$ since this ball does not contain any point of C_i (see top-left figure in Figure 3, where $C_i = C_5$ and $C_j = C_8$). Furthermore, we end up deleting at least one cluster (specifically, C_j), so this process of temporarily deleting balls can repeat at most k times before C_i intersects the extended ball D'_i. Once we have a good dense ball $D_i = \mathsf{ball}(y_i, r_i)$ for C_i, consider the extended ball $D'_i := \mathsf{ball}(y_i, r_i + 2r_j)$, where r_j is the radius of the anchor C_j for D_i. There are two cases:

Case 1: Hop through the anchors. Suppose we are lucky, and it turns out that $r_j \leq r_i$. Then, y_i can serve all the points of C_i within radius $3r_i + 2r_j \leq 5r_i$. See top-right figure in Figure 3, where $C_j = C_3$. We now mark C_i as processed (color black). Note that, in this case, we do not delete any points.

Case 2: Otherwise "Exchange". Suppose we are not lucky, and it turns out that $r_j > r_i$ (see bottom-left figure in Figure 3 where $r_i = r_5 > r_8 = r_j$). In this case, our basic approach is to open the cluster C_i at D_i, and thus relieving the clusters intersecting D_i from their responsibility for these points. In exchange, these clusters will become responsible for C_i points. The crux of the argument is that as the anchor C_j has the minimal radius among the clusters intersecting D_i, each such cluster C_q has radius larger than r_i. Thus from C_q's perspective, the points of C_i are "nearby". Hence C_q can accept responsibility for a number of C_i points proportional to the number of C_q points taken (and thus everybody is taken care of).

In more details: we create a new cluster $C'_i \subseteq D_i$ out of the points in D_i (recall $|D_i| \geq |C_i|$ and see Footnote 11). To make this assignment permanent, we delete D_i, and mark C_i as processed (color black). However, we end up taking points from other clusters (and also end up deleting points from these clusters). The key observation is that as we process the clusters in non-decreasing order and $r_j > r_i$, D_i does not contain points from clusters processed via Case 1. Since we maintain the invariant that the points of unprocessed clusters are not deleted, we have to mark the clusters intersecting D_i as processed (color black). However, before marking these clusters as processed, we need to ensure that they are accounted for in our solution. To this end, we use a novel idea of exchanging points. Since we have created a new cluster C'_i out of D_i, any cluster $C_{j'}$ that has lost, say, $n_{j'} > 0$ points in this process, can instead claim back $n_{j'}$ points from the original cluster C_i (depicted by islands in C_5 in bottom-left figure in Figure 3) by paying slightly more cost since, in this case, we have $r_i < r_j \leq r_{j'}$. Here, we use the fact that C_j is the anchor for D_i, and hence r_j is the smallest radius intersecting D_i. We call such clusters partitioned clusters. Specifically, the radius of

the partitioned $C_{j'}$ is at most $\delta(o_{j'}, y_i) + \delta(y_i, p) \leq (r_{j'} + r_i) + (r_i + 2r_j + 2r_i) \leq 7r_{j'}$, for $p \in C_i$, as $r_j \leq r_{j'}$. See bottom-left figure in for an illustration. Note that a cluster $C_{j'}$ can be partitioned multiple times by different C_i's (we track the clusters that partitioned $C_{j'}$ by $\mathsf{Partitions\text{-}of}(C_{j'})$), as shown in bottom-right figure in Figure 3. However, $o_{j'}$ can still serve all the points of the modified $C_{j'}$ within cost $7r_{j'}$, since, in this case, the radii of the clusters partitioning $C_{j'}$ are strictly less than $r_{j'}$. While it is hard to find $o_{j'}$, in the full version, we show how to find another point that can serve all the points of the partitioned cluster $C_{j'}$ within radius $7r_{j'}$.

Note that, we get different approximation factors in Case 1 and Case 2. In the technical section, we interpolate between these two factors to obtain an improved approximation factor for the algorithm.

2.2 Uniform capacities

Our algorithm builds upon the algorithm of [58]. The algorithm divides the clusters in $\mathcal{C}$ into two categories: heavy and light. A cluster $C \in \mathcal{C}$ is *heavy* if $|C| > U/k^3$; otherwise, it is a *light* cluster. The intuition behind this partition is the following: some heavy cluster must exist, and given such heavy cluster $C \in \mathcal{C}$, a randomly selected point from P belongs to C with probability at least $\text{poly}(1/k)$. Therefore, with probability at least $k^{-O(k)}$, we can obtain a set $X \subseteq P$ containing a single point x_i from every heavy cluster $C_i \in \mathcal{C}$. For every heavy cluster C_i, we wish to open a cluster $B_i = \mathsf{ball}(x_i, 2r_i)$ that will take care of C_i points (note $C_i \subseteq B_i$). We will assume that the union of these balls, $\cup_{x_i \in X} B_i$, covers the point set P (this assumption catches the essence of the problem, as we can greedily take care of light clusters not covered by this union). Let $\mathcal{L} \subseteq P$ be the set of points corresponding to the light clusters in $\mathcal{C}$. For each light cluster $C_t \in \mathcal{C}$, arbitrarily assign it to a heavy cluster $C_i \in \mathcal{C}$ such that C_t intersects B_i. Then, for each heavy cluster C_i, consider the set $S_i := C_i \cup (\mathsf{ball}(x_i, 2r_i + 2r_{\ell(i)}) \cap \mathcal{L})$, where $r_{\ell(i)}$ is the maximum radius of a light cluster assigned to C_i. Note that S_i contains C_i, and all the light clusters assigned to C_i. If $|S_i| \leq U$, then we can open cluster $C'_i = S_i$ without violating the uniform capacity. For the other case, note that $|S_i| \leq (1 + 1/k^2)U$ since there can be at most k light clusters assigned to C_i, each of which has at most U/k^3 points. Let $I \subseteq [k]$ be the indices of heavy clusters, and let $I' \subseteq I$ be the indices of heavy clusters for whom S_i violated the capacity constraint. [58] used a very neat matching based argument to construct $|I'|$ balls $\{\hat{C}_i\}_{i \in I'}$ each with a unique radius from $\{r_i\}_{i \in I}$ (the heavy clusters) such that $\cup_{i \in I'} \hat{C}_i$ contains at least U/k^2 points from each S_i (for $i \in I'$). These clusters $\{\hat{C}_i\}_{i \in I'}$ are then used to unload enough points from the overloaded heavy clusters. Now, $\{\mathsf{ball}(x_i, 2r_i + 2r_{\ell(i)})\}_{i \in I}$ together with $\{\hat{C}_i\}_{i \in I'}$ is a valid feasible solution. The ℓ_1 norm cost (CapSoR cost) of this solution is bounded by $\sum_{i \in I}(2r_i + 2r_{\ell(i)}) + \sum_{j \in I} r_j \leq 3 \sum_{i \in [k]} r_i \leq (3 + \varepsilon)\text{OPT}$, where we used that fact that $C_{\ell(i)}$ is distinct for each C_i, and that in $\cup_{i \in I'} \hat{C}_i$ we used the radius of every heavy cluster at most once. On the other hand, the ℓ_∞ cost (Capacitated k-Center) can only be bounded by $(4 + \varepsilon)\text{OPT}$. It is also possible to bound the approximation factor for general ℓ_p norm objective.

Improvement for ℓ_p norm objectives

Our modification to get a better factor is very small, but leads to a significant improvement. Specifically, we fine tune the definition of $C_{\ell(i)}$ for C_i. [58] assigned each light cluster C_j to an arbitrary heavy cluster C_i such that C_j intersects the extended ball B_i, and let $r_{\ell(i)}$ to be the maximum radius of a light cluster assigned to C_i. Instead, we assign each light

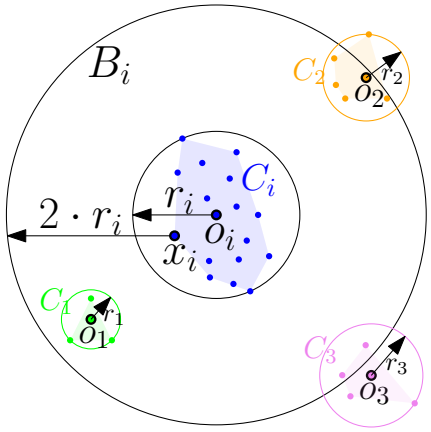


Figure 2 The sampled point x_i belongs to the heavy cluster, C_i.

cluster C_j to an arbitrary heavy cluster C_i such that B_i contains its center o_i, while $r_{\ell(i)}$ remains the maximum radius of a light cluster assigned to C_i. The crux is that now the ball $\mathsf{ball}(x_i, 2r_i + r_{\ell(i)})$ contains the heavy cluster C_i and all the light clusters assigned to it. In the example shown in Figure 2, we consider a heavy cluster C_i, where we sampled a point $x_i \in C_i$ and consider the ball $B_i = \mathsf{ball}(x_i, 2r_i)$. B_i intersects 3 light clusters (C_1, C_2, C_3) with radii $r_1 < r_2 < r_3$. Now $r_{\ell(i)} = r_2$ because $o_2 \in B_i$, while $o_3 \notin B_i$. As previously, we open clusters of two types: (1) for every heavy cluster C_i, a cluster C'_i centered at x_i of radius $2r_i + r_{\ell(i)}$, and (2) the clusters $\{\hat{C}\}_{i \in I'}$ relieving the extra load. Overall, we save a factor of $r_{\ell(i)}$ in the radius of the cluster centered at x_i. Note that this solution yields a 3 approximation w.r.t. the ℓ_∞ norm objective (Capacitated k-Center), and yields a similar improvement w.r.t. other ℓ_p norm objectives.

2.3 Hardness of FPT-approximation for uniform capacities

Our hardness result is based on a polynomial time reduction from Max k-Coverage (Max k-Cov), where, given a universe $\mathcal{G}$ of elements, a positive integer k, and k collections $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ of subsets of G, the task is to find $\{S_1, \ldots, S_k\}$ such that $S_i \in \mathcal{C}_i, i \in [k]$ that maximizes the number of elements covered. It is known that Max k-Cov is NP-hard to approximate to a factor better than $(1 - 1/e)$, even when every set in the collection has precisely $|\mathcal{G}|/k$ elements. Under gap-ETH, there is no $\mathsf{FPT}(k)$ algorithm for Max k-Cov that approximates to a factor better than $(1 - 1/e)$, even when every set in the collection has precisely $|\mathcal{G}|/k$ elements

On a high level, given an instance of Max k-Cov, we create the set-element incidence graph of the instance. We then add edges between the vertices corresponding to the sets of the same collection. Let $G = (V, E)$ be the resulting graph. The points of our CapSoR instance corresponds to the vertices in V, and the distance metric is given by the shortest path distance in G. Finally, we set the uniform capacity $U = |\mathcal{G}|/k + m$. It is easy to see that when there are k sets $\{S_1, \ldots, S_k\}$ such that $S_i \in \mathcal{C}_i, i \in [k]$ that cover $\mathcal{G}$, then we can find a solution for CapSoR with cost k, by selecting the points corresponding to the sets $\{S_1, \ldots, S_k\}$ as the centers. For the No case, observe that any solution to CapSoR must have clusters of size precisely U due to our construction. This simple observation forces the centers to belong to different collections, yielding the result.

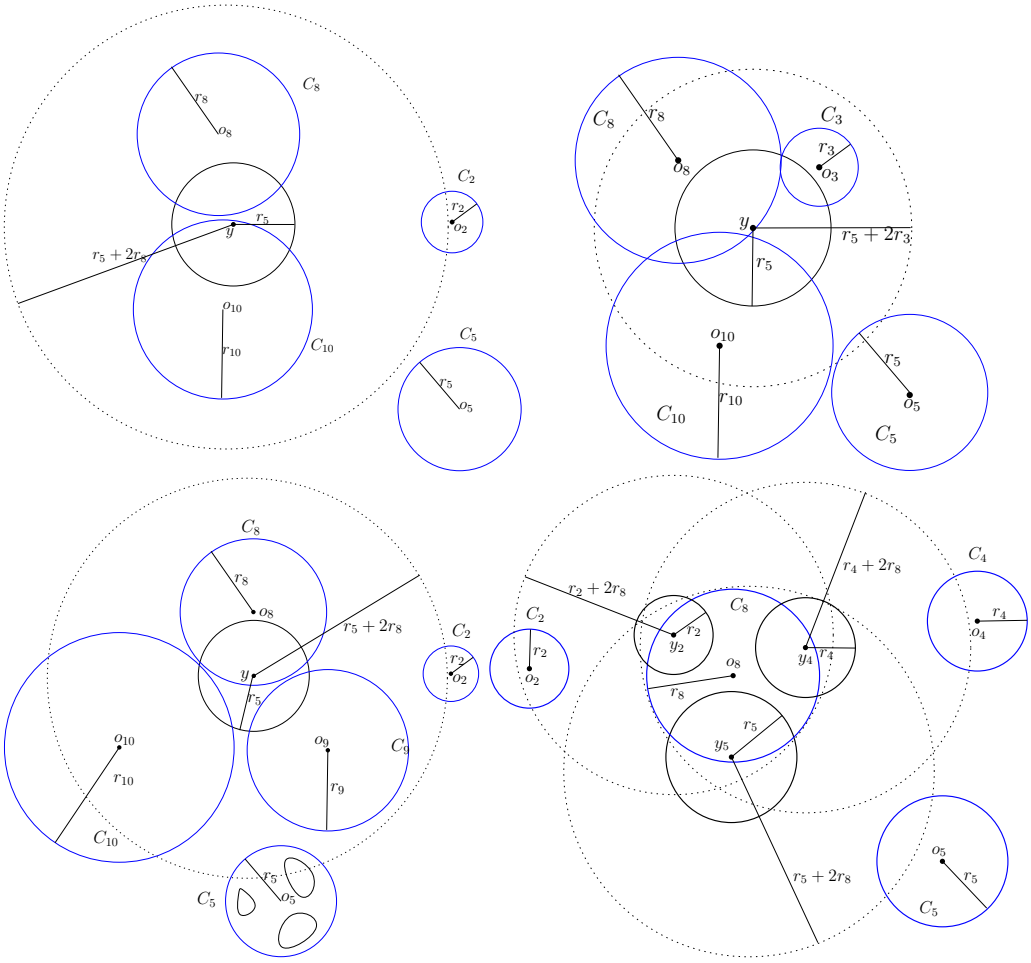


Figure 3 Top-Left figure shows a dense ball $\mathsf{ball}(y, r_5)$, colored in black, that is not good for cluster C_5, because $\mathsf{ball}(y, r_5 + 2r_8)$ is disjoint from C_5. Hence, it is safe to (temporary) delete $\mathsf{ball}(y, r_5 + 2r_8)$ without affecting cluster C_5, and recursively search for good dense ball for C_5 in the remaining points. On the other hand, the $\mathsf{ball}(y, r_5)$ of Top-Right figure is good dense ball for C_5. Furthermore, note that in this case the radius of the anchor $r_3 \leq r_5$, and hence y can serve all the points of C_5 within distance $3r_5 + 2r_3 \leq 5r_5$. In Bottom-left figure, the radius of the anchor is $r_8 > r_5$, and hence the previous argument fails. However, in this case, since $\mathsf{ball}(y,r_5)$ is a good dense ball for C_5, so we can open a new cluster $C_5' \subseteq \mathsf{ball}(y, r_5)$ of appropriate size, instead of serving the points of C_5. Note that, in this case, the new cluster C_5' ends up taking points from clusters C_8, C_9 and C_{10}. However, these clusters can claim back the lost points from the original cluster C_5 (depicted by islands in C_5), which is not too far from them. In this case, we say that C_8, C_9 and C_{10} are partitioned by C_5'. Bottom-right figure shows cluster C_8 partitioned by C_2, C_4 and C_5, and it has to collect its lost points from the respective clusters.

3 Further related work

We begin with further background on the (uncapacitated) Sum of Radii and Sum of Diameters problems. When the point set P is from either ℓ_1 or ℓ_2 there is an exact poly-time algorithm [45], while there is also an EPTAS for the Euclidean (ℓ_2) case [45]. Another interesting case is when the metric between the input points P corresponds to the shortest path metric of an unweighted graph G. Here, assuming the optimal solution does not contain singletons

(clusters with only a single point), there is a polynomial time algorithm finding optimal solution [14]. The Sum of Diameters with fixed k admits exact polytime algorithm [14] (see also [51, 26]). Further, there is a PTAS for Sum of Diameters where the input points are from the Euclidean plane [14]. The Sum of Radii problem was also studied in the fault tolerant regime [17], considering outliers [3], and in an online (competitive analysis) setting [42, 36].

Capacitated k-Center is a special case of capacitated ℓ_p norm of radii when $p = \infty$. The uniform Capacitated k-Center problem has been explicitly studied by Goyal and Jaiswal [50], and is also implicitly captured by the results of [58]. Goyal and Jaiswal design a factor 2 FPT-approximation for the soft assignment setting, where multiple centers can be opened at a single point, enabling it to serve more than U points. For instance, the algorithm can opt to open two centers at a point $p \in P$, thereby allowing up to $2U$ points to be served from location p. In contrast, similar to this paper, [58] focus on the hard assignment scenario, where at most one center can be opened at a given point, and they present a $(4 + \varepsilon)$ FPT-approximation for this case. We are not aware of any approximation preserving subroutines that transform a soft assignment algorithm to a hard assignment algorithm. However, by replacing the chosen center with any other point in the cluster one can obtain a solution with hard assignments while incurring a factor 2 multiplicative loss in approximation. Consequently, the algorithm of [50] can only yield 4 FPT-approximation using this approach, which matches the factor of [58].

Capacitated Max-k-Diameter is a special case of capacitated ℓ_p norm of diameters problem when $p = \infty$. Its uncapacitated version turns out to be computationally very challenging. While there is a simple 2-approximation for Max-k-Diameter [47], it is NP-hard to approximate the problem better than factor 2 in ℓ_∞ metric even for $k = 3$ [69]. Very recently, Fleischmann [41] showed that, even for $k = 3$, the problem is NP-hard to approximate better than factors 1.5 and 1.304 when the points come from ℓ_1 and ℓ_2 metrics, respectively. Given that these hardness results hold for constant k (i.e., even for $k = 3$), they also translate to FPT lower bounds. Thus, these hardness results imply that no FPT algorithm can beat these lower bounds for Max-k-Diameter, unless P = NP. Of course, the corresponding capacitated version of Max-k-Diameter is a generalization of Max-k-Diameter, and hence these FPT-hardness of approximations also hold for the capacitated version. For upper bounds, it is interesting to note that the reduction from Sum of Radii to Sum of Diameters mentioned in Footnote 10 works even when solving Sum of Radii w.r.t. soft assignments. Thus using [50] one can obtain a 4 FPT-approximation for the uniform Capacitated Max-k-Diameter problem, matching our Theorem 5 for this case.

References

1 Fateme Abbasi, Sandip Banerjee, Jarosław Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. Parameterized approximation schemes for clustering with general norm objectives. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1377–1399, 2023. doi:10.1109/FOCS57990.2023.00085.

2 Marek Adamczyk, Jaroslaw Byrka, Jan Marcinkowski, Syed Mohammad Meesum, and Michal Wlodarczyk. Constant-factor FPT approximation for capacitated k-median. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 1:1–1:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.1.

3 Sara Ahmadian and Chaitanya Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 69:1–69:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.69.

4 Helmut Alt, Esther M Arkin, Hervé Brönnimann, Jeff Erickson, Sándor P Fekete, Christian Knauer, Jonathan Lenchner, Joseph SB Mitchell, and Kim Whittlesey. Minimum-cost coverage of point sets by disks. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 449–458, 2006.

5 Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. *Math. Program.*, 154(1-2):29–53, 2015. doi:10.1007/S10107-014-0857-Y.

6 Barbara Anthony, Vineet Goyal, Anupam Gupta, and Viswanath Nagarajan. A plant location guide for the unsure: Approximation algorithms for min-max location problems. *Mathematics of Operations Research*, 35(1):pages 79–101, 2010.

7 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004. doi:10.1137/S0097539702416402.

8 Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The Hardness of Approximation of Euclidean k-Means. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 754–767, Dagstuhl, Germany, 2015. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SOCG.2015.754.

9 Mihai Badŏiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In John H. Reif, editor, *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC'02)*, pages 250–257. ACM, 2002. doi:10.1145/509907.509947.

10 Tanvi Bajpai, Deeparnab Chakrabarty, Chandra Chekuri, and Maryam Negahbani. Revisiting Priority k-Center: Fairness and Outliers. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.21.

11 Sayan Bandyapadhyay, William Lochet, and Saket Saurabh. FPT Constant-Approximations for Capacitated Clustering to Minimize the Sum of Cluster Radii. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:14, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2023.12.

12 Sayan Bandyapadhyay and Kasturi R. Varadarajan. Approximate clustering via metric partitioning. In Seok-Hee Hong, editor, *27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia*, volume 64 of *LIPIcs*, pages 15:1–15:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ISAAC.2016.15.

13 Sandip Banerjee, Yair Bartal, Lee-Ad Gottlieb, and Alon Hovav. Novel properties of hierarchical probabilistic partitions and their algorithmic applications. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1724–1767, 2024. doi:10.1109/FOCS61266.2024.00107.

14 Babak Behsaz and Mohammad R. Salavatipour. On minimum sum of radii and diameters clustering. *Algorithmica*, 73(1):143–165, 2015. doi:10.1007/S00453-014-9907-3.

15 Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Faster algorithms for the constrained k-means problem. *Theory of Computing Systems*, 62(1):93–115, 2018. doi:10.1007/S00224-017-9820-7.

16 Sayan Bhattacharya, Parinya Chalermsook, Kurt Mehlhorn, and Adrian Neumann. New approximability results for the robust k-median problem. In *Scandinavian Workshop on Algorithm Theory (SWAT'14)*, pages 50–61. Springer, 2014. doi:10.1007/978-3-319-08404-6_5.

17 Santanu Bhowmick, Tanmay Inamdar, and Kasturi R. Varadarajan. Fault-tolerant covering problems in metric spaces. *Algorithmica*, 83(2):413–446, 2021. doi:10.1007/S00453-020-00762-Y.

18 Vittorio Bilò, Ioannis Caragiannis, Christos Kaklamanis, and Panagiotis Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In Gerth Stølting Brodal and Stefano Leonardi, editors, *Algorithms - ESA 2005, 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005, Proceedings*, volume 3669 of *Lecture Notes in Computer Science*, pages 460–471. Springer, 2005. doi:10.1007/11561071_42.

19 Vittorio Bilò, Ioannis Caragiannis, Christos Kaklamanis, and Panagiotis Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In Gerth Stølting Brodal and Stefano Leonardi, editors, *Algorithms – ESA 2005*, pages 460–471, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11561071_42.

20 Vladimir Braverman, Shaofeng H-C Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for ordered weighted clustering. In *International Conference on Machine Learning (ICML'19)*, pages 744–753. PMLR, 2019. URL: http://proceedings.mlr.press/v97/braverman19a.html.

21 Moritz Buchem, Katja Ettmayr, Hugo K. K. Rosado, and Andreas Wiese. A $(3+\varepsilon)$-approximation algorithm for the minimum sum of radii problem with outliers and extensions for generalized lower bounds. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 1738–1765. SIAM, 2024. doi:10.1137/1.9781611977912.69.

22 Jaroslaw Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 722–736. SIAM, 2015. doi:10.1137/1.9781611973730.49.

23 Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2):23:1–23:31, 2017. doi:10.1145/2981561.

24 Jaroslaw Byrka, Bartosz Rybicki, and Sumedha Uniyal. An approximation algorithm for uniform capacitated k-median problem with 1+\epsilon capacity violation. In Quentin Louveaux and Martin Skutella, editors, *Integer Programming and Combinatorial Optimization - 18th International Conference, IPCO 2016, Liège, Belgium, June 1-3, 2016, Proceedings*, volume 9682 of *Lecture Notes in Computer Science*, pages 262–274. Springer, 2016. doi:10.1007/978-3-319-33461-5_22.

25 Jarosław Byrka, Krzysztof Sornat, and Joachim Spoerhase. Constant-factor approximation for ordered k median. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'18)*, pages 620–631, 2018. doi:10.1145/3188745.3188930.

26 Vasilis Capoyleas, Günter Rote, and Gerhard J. Woeginger. Geometric clusterings. *J. Algorithms*, 12(2):341–356, 1991. doi:10.1016/0196-6774(91)90007-L.

27 Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 126–137, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316322.

28 Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *Journal of Computer and System Sciences*, 68(2):417–441, 2004. Special Issue on STOC 2001. doi:10.1016/j.jcss.2003.07.014.

29 Xianrun Chen, Dachuan Xu, Yicheng Xu, and Yong Zhang. Parameterized approximation algorithms for sum of radii clustering and variants. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 20666–20673. AAAI Press, 2024. doi:10.1609/AAAI.V38I18.30053.

30 Julia Chuzhoy and Yuval Rabani. Approximating k-median with non-uniform capacities. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 952–958. SIAM, 2005. URL: http://dl.acm.org/citation.cfm?id=1070432.1070569.

31 Vincent Cohen-Addad, Anupam Gupta, Lunjia Hu, Hoon Oh, and David Saulpic. An improved local search algorithm for k-median. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1556–1612. SIAM, 2022. doi:10.1137/1.9781611977073.65.

32 Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT approximations for k-median and k-means. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.42.

33 Vincent Cohen-Addad and Karthik C. S. Inapproximability of clustering in lp metrics. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 519–539. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00040.

34 Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. Johnson coverage hypothesis: Inapproximability of k-means and k-median in ℓ_p-metrics. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1493–1530. SIAM, 2022. doi:10.1137/1.9781611977073.63.

35 Vincent Cohen-Addad and Jason Li. On the fixed-parameter tractability of capacitated clustering. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 41:1–41:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.41.

36 János Csirik, Leah Epstein, Csanád Imreh, and Asaf Levin. Online clustering with variable sized clusters. *Algorithmica*, 65(2):251–274, 2013. doi:10.1007/S00453-011-9586-2.

37 Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. LP rounding for k-centers with non-uniform hard capacities. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 273–282. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.63.

38 Sanjoy Dasgupta. The hardness of k-means clustering. Technical Report CS2008-0916, University of California, San Diego, San Diego, CA, 2008.

39 H. Gökalp Demirci and Shi Li. Constant approximation for capacitated k-median with (1+epsilon)-capacity violation. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 73:1–73:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.73.

40 Srinivas Doddi, Madhav V. Marathe, S. S. Ravi, David Scot Taylor, and Peter Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. *Nordic J. of Computing*, 7(3):185–203, September 2000.

41 Henry L. Fleischmann, Kyrylo Karlov, Karthik C. S., Ashwin Padaki, and Stepan Zharkov. Inapproximability of maximum diameter clustering for few clusters. *CoRR*, abs/2312.02097, 2023. doi:10.48550/arXiv.2312.02097.

42 Dimitris Fotakis and Paraschos Koutris. Online sum-radii clustering. *Theor. Comput. Sci.*, 540:27–39, 2014. doi:10.1016/J.TCS.2013.03.010.

43 Zachary Friggstad and Mahya Jamshidian. Improved Polynomial-Time Approximations for Clustering with Minimum Sum of Radii or Diameters. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 56:1–56:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2022.56.

44 Mehrdad Ghadiri, Samira Samadi, and Santosh Vempala. Socially fair k-means clustering. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 438–448, 2021. doi:10.1145/3442188.3445906.

45 Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A. Pirwani, and Kasturi Varadarajan. On metric clustering to minimize the sum of radii. *Algorithmica*, 57(3):484–498, July 2010. doi:10.1007/S00453-009-9282-7.

46 Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A. Pirwani, and Kasturi R. Varadarajan. On clustering to minimize the sum of radii. *SIAM J. Comput.*, 41(1):47–60, 2012. doi:10.1137/100798144.

47 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

48 Inge Li Gørtz and Anthony Wirth. Asymmetry in k-center variants. *Theor. Comput. Sci.*, 361(2-3):188–199, 2006. doi:10.1016/J.TCS.2006.05.009.

49 Kishen N. Gowda, Thomas W. Pensyl, Aravind Srinivasan, and Khoa Trinh. Improved bi-point rounding algorithms and a golden barrier for k-median. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 987–1011. SIAM, 2023. doi:10.1137/1.9781611977554.CH38.

50 Dishant Goyal and Ragesh Jaiswal. Tight fpt approximation for constrained k-center and k-supplier. *Theoretical Computer Science*, 940:190–208, 2023. doi:10.1016/j.tcs.2022.11.001.

51 Pierre Hansen and Brigitte Jaumard. Minimum sum of diameters clustering. *Journal of Classification*, 4(2):215–226, 1987.

52 Pierre Hansen and Brigitte Jaumard. Cluster analysis and mathematical programming. *Math. Program.*, 79:191–215, 1997. doi:10.1007/BF02614317.

53 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 291–300. ACM, 2004. doi:10.1145/1007352.1007400.

54 Martin G. Herold, Evangelos Kipouridis, and Joachim Spoerhase. Clustering to minimize cluster-aware norm objectives. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 255–287. SIAM, 2025. doi:10.1137/1.9781611978322.8.

55 Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985. doi:10.1287/MOOR.10.2.180.

56 Tanmay Inamdar and Kasturi Varadarajan. Capacitated Sum-Of-Radii Clustering: An FPT Approximation. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2020.62.

57 Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 731–740. ACM, 2002. doi:10.1145/509907.510012.

58 Ragesh Jaiswal, Amit Kumar, and Jatin Yadav. FPT Approximation for Capacitated Sum of Radii. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, volume 287 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 65:1–65:21, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2024.65.

59 Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004. doi:10.1016/J.COMGEO.2004.03.003.

60 Samir Khuller and Yoram J. Sussmann. The capacitated K-center problem. *SIAM J. Discret. Math.*, 13(3):403–418, 2000. doi:10.1137/S0895480197329776.

61 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J.ACM*, 57(2):1–32, 2010. doi:10.1145/1667053.1667054.

62 G. Laporte, S. Nickel, and F. S. da Gama. *Location Science*. Springer, 2015.

63 Shi Li. Approximating capacitated k-median with $(1 + \varepsilon)k$ open facilities. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 786–796. SIAM, 2016. doi:10.1137/1.9781611974331.CH56.

64 Shi Li. On uniform capacitated k-median beyond the natural LP relaxation. *ACM Trans. Algorithms*, 13(2):22:1–22:18, 2017. doi:10.1145/2983633.

65 Xiaofei Liu, Weidong Li, and Runtao Xie. A primal-dual approximation algorithm for the k-prize-collecting minimum power cover problem. *Optimization Letters*, 16(8):2373–2385, 2022. doi:10.1007/S11590-021-01831-Z.

66 S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, March 1982. doi:10.1109/TIT.1982.1056489.

67 Yury Makarychev and Ali Vakilian. Approximation algorithms for socially fair clustering. In *Conference on Learning Theory (COLT'21)*, pages 3246–3264. PMLR, 2021. URL: http://proceedings.mlr.press/v134/makarychev21a.html.

68 Jirı Matoušek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84, 2000. doi:10.1007/S004540010019.

69 Nimrod Megiddo. On the complexity of some geometric problems in unbounded dimension. *Journal of Symbolic Computation*, 10(3):327–334, 1990. doi:10.1016/S0747-7171(08)80067-3.

70 Clyde Monma and Subhash Suri. Partitioning points and graphs to minimize the maximum or the sum of diameters. In *Graph Theory, Combinatorics and Applications (Proc. 6th Internat. Conf. Theory Appl. Graphs)*, volume 2, pages 899–912, 1989.

71 S. Nickel and J. Puerto. *Location Theory*. Springer Science & Business Media, 2005.

72 Ján Plesník. A heuristic for the p-center problems in graphs. *Discrete Applied Mathematics*, 17(3):263–268, 1987. doi:10.1016/0166-218X(87)90029-1.

73 Yicheng Xu, Yong Zhang, and Yifei Zou. A constant parameterized approximation for hard-capacitated k-means. *CoRR*, abs/1901.04628, 2019. arXiv:1901.04628.

Algorithms for Euclidean Distance Matrix Completion: Exploiting Proximity to Triviality

Fedor V. Fomin
University of Bergen, Norway

Petr A. Golovach
University of Bergen, Norway

M. S. Ramanujan
University of Warwick, UK

Saket Saurabh
Institute of Mathematical Sciences, Chennai, India
University of Bergen, Norway

Abstract

In the d-Euclidean Distance Matrix Completion (d-EDMC) problem, one aims to determine whether a given partial matrix of pairwise distances can be extended to a full Euclidean distance matrix in d dimensions. This problem is a cornerstone of computational geometry with numerous applications. While classical work on this problem often focuses on exploiting connections to semidefinite programming typically leading to approximation algorithms, we focus on exact algorithms and propose a novel *distance-from-triviality parameterization* framework to obtain tractability results for d-EDMC.

We identify key structural patterns in the input that capture entry density, including chordal substructures and coverability of specified entries by fully specified principal submatrices. We obtain:

1. The first fixed-parameter algorithm (FPT algorithm) for d-EDMC parameterized by d and the maximum number of unspecified entries per row/column. This is achieved through a novel *compression algorithm* that reduces a given instance to a submatrix on $\mathcal{O}(1)$ rows (for fixed values of the parameters).
2. The first FPT algorithm for d-EDMC parameterized by d and the minimum number of fully specified principal submatrices whose entries cover all specified entries of the given matrix. This result is also achieved through a compression algorithm.
3. A polynomial-time algorithm for d-EDMC when both d and the minimum fill-in of a natural graph representing the specified entries are fixed constants. This result is achieved by combining tools from distance geometry and algorithms from real algebraic geometry.

Our work identifies interesting parallels between EDM completion and graph problems, with our algorithms exploiting techniques from both domains.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms; Theory of computation → Fixed parameter tractability

Keywords and phrases Parameterized Complexity, Euclidean Embedding, Polynomial Compression

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.49

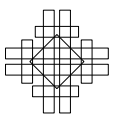
Related Version *Full Version*: `https://arxiv.org/abs/2603.19447`

Funding *Fedor V. Fomin*: Supported by the Research Council of Norway under BWCA project (grant no. 314528) and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (NewPC grant agreement No. 101199930)
Petr A. Golovach: Supported by the Research Council of Norway under the BWCA (grant no. 314528) and Extreme-Algorithms (grant no 355137) projects.
M. S. Ramanujan: Supported by Engineering and Physical Sciences Research Council (EPSRC) grant EP/V044621/1.



© Fedor V. Fomin, Petr A. Golovach, M. S. Ramanujan, and Saket Saurabh;
licensed under Creative Commons License CC-BY 4.0





Funded by the European Union

Saket Saurabh: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416); and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

1 Introduction

Consider the following problem: given an $n \times n$ symmetric hollow matrix M, and integer d, determine whether M is a Euclidean Distance Matrix (EDM) that is isometrically embeddable into $\mathbb{R}^d$, also said to be *realizable in* $\mathbb{R}^d$. In other words, determine whether there exists a set $P = \{p_1, \ldots, p_n\}$ of n points in $\mathbb{R}^d$ such that $M_{ij} = \|p_i - p_j\|_2^2$, where M_{ij} denotes the (i,j)-th entry of M, and $\|p_i - p_j\|_2$ is the Euclidean distance between p_i and p_j. This question of determining the existence of such a point configuration P (up to congruence) is a central problem in Distance Geometry, whose solution is well understood and dates back to the works of Cayley [13] and Menger [34].

In this paper, we are interested in the version of this problem where the matrix M is *partial*, i.e., some entries may be "unspecified." In this version, called d-EDM Completion (d-EDMC), the goal is to decide whether the missing entries in a matrix can be filled in so that the result is an EDM that can be realized in $\mathbb{R}^d$. In general, such questions of reconstructing point configurations from *partial* distance measurements naturally occur across numerous disciplines. The study of d-EDMC in particular, goes back to mathematicians and mechanical engineers of the nineteenth century – see, for example, the work of Cauchy from 1813 [12] – and is at the heart of rigidity theory. This problem has many applications in different areas, such as molecular conformation in bioinformatics [31, 15, 14], dimensionality reduction in machine learning and statistics, and localization in wireless sensor networks [3, 21, 9].

Saxe [35] and Yemini [40] showed that the d-EDMC problem is already *strongly* NP-hard in dimensions $d = 1$ and $d = 2$. While numerous heuristics for solving d-EDMC under specific structural assumptions on M have been proposed in the literature [23, 39, 8, 11], *theoretical* results with provable performance guarantees are scarce. This lack of rigorous algorithmic understanding of the problem is a primary motivation for us to explore the computational complexity of this problem and in particular, its parameterized complexity.

In this work we adopt the *distance-from-triviality* viewpoint to study the complextity of d-EDMC. Distance-from-triviality is a well-established paradigm in parameterized complexity, where the idea is to introduce a parameter (or a combination of parameters) that measures how far a given instance lies from a class that can be solved efficiently; algorithms then exploit this "distance" to achieve tractability.

What is the notion of "triviality" for d-EDMC*?* Two natural extremal cases suggest themselves.

1. *All entries are unspecified.* If no entry of the matrix is given, *any* Euclidean distance matrix of the appropriate size is a valid completion, so the instance is trivial. Unfortunately, even a very small deviation from this case already yields hardness: Saxe's reduction [35] shows NP-hardness when each row and column contains *exactly two* specified entries. Consequently, parameterizing by distance to the completely unspecified matrix appears unpromising.

2. *All entries are specified.* At the opposite extreme, when every entry is given, one need only test whether the matrix is an EDM that embeds isometrically into $\mathbb{R}^d$, a task solvable in polynomial time. As we show in this paper, distance to this notion of "triviality" is a much more interesting parameter.

We may therefore formulate our main question as follows.

> Which notions of distance to a fully specified matrix can be exploited algorithmically for d-EDMC?

From the standpoint of algorithms with provable guarantees, this question is largely unexplored. A notable exception is the work of Berger, Kleinberg, and Leighton [7]. They show that if, in every row of an $n \times n$ partial matrix M, at least $3n/4+1$ entries are specified, then one can decide in polynomial time whether M can be completed to a 3-EDM. There is, however, a crucial caveat: their theorem assumes that the target point set is in *general position* and that no ten points lie on a quadric surface. We make no such assumptions – the points to be embedded need not be in general position or even distinct – rendering the problem substantially more realistic but also more challenging.

1.1 Our results and methods

We begin by observing that high density of specified entries (or, equivalently, a numerical sparsity of unspecified entries) does not by itself make the problem any easier. To exploit sparsity algorithmically, one must therefore identify and leverage additional *structural properties* that reflect the density of entries.

▶ **Theorem 1.** *$\forall \varepsilon \in (0,1)$, d-EDMC remains strongly* NP*-hard even for instances (M,d) with $d=2$ in which the $n \times n$ matrix M contains at most εn unspecified entries.*

The proof of the theorem is an adaptation of the classic complexity result of Saxe [35]. It also underscores that the general-position assumptions employed by Berger, Kleinberg, and Leighton [7] is essential for their polynomial-time algorithm.

1.1.1 Distance from triviality I

Our first main algorithmic result shows that imposing even a mild structural constraint on the unspecified entries – beyond mere numerical sparsity – makes the problem tractable. Our notion of density of a matrix is expressed by *forbidden t-block patterns*.

▶ **Definition 2** (t-block pattern)**.** For an $n \times n$ matrix M and disjoint sets $I_1, I_2 \subseteq [n]$, we define by $M[I_1, I_2]$ the submatrix of M indexed by the rows in I_1 and the columns in I_2. We say that $M[I_1, I_2]$ is a *t-block pattern* if it is a $t \times t$ submatrix of M where every entry is unspecified. We say that M *excludes* a t-block pattern if there is no I_1 and I_2 such that $M[I_1, I_2]$ is a t-block pattern. See top of Figure 1.

▶ **Example 3.** Consider a partial matrix M. M excludes a 1-block pattern if and only if it is fully specified; if M has at most Δ unspecified entries in total, then M excludes a $\lfloor\sqrt{\Delta+1}\rfloor$-block pattern; if each row of M contains at most Δ unspecified entries, then M excludes a $(\Delta+1)$-block pattern.

We are now ready to state our first tractability result. By *solving* an instance of d-EDMC, we mean deciding[1] whether the answer is yes or no. We also say that an instance (M,d) is *equivalent* to an instance (M',d') if solving both instances leads to the same answer.

▶ **Theorem 4** (Compression for $K_{t,t}$-free complements)**.** *There is a polynomial-time algorithm that, given an instance (M,d) of d-EDMC in which M excludes a t-block pattern for some $t \geq 1$, either solves the problem, or outputs an equivalent instance (M',d), such that M' is a principal submatrix of M with $(d+1)^{\mathcal{O}(t^2)}$ rows and columns.*

[1] As is standard in Computational Geometry, all our algorithms operate under the *real RAM* computational model, assuming that basic operations over real numbers can be executed in unit time.

$$\begin{bmatrix} 0 & * & * & * & * & 2 & * & * & * \\ * & 0 & 1 & * & * & * & * & * & * \\ * & 1 & 0 & * & * & * & * & 2 & * \\ * & * & * & 0 & 1 & * & * & * & 2 \\ * & * & * & 1 & 0 & 1 & * & * & * \\ 2 & * & * & * & 1 & 0 & 1 & * & * \\ * & * & * & * & * & 1 & 0 & * & * \\ * & * & 2 & * & * & * & * & 0 & * \\ * & * & * & 2 & * & * & * & * & 0 \end{bmatrix}$$

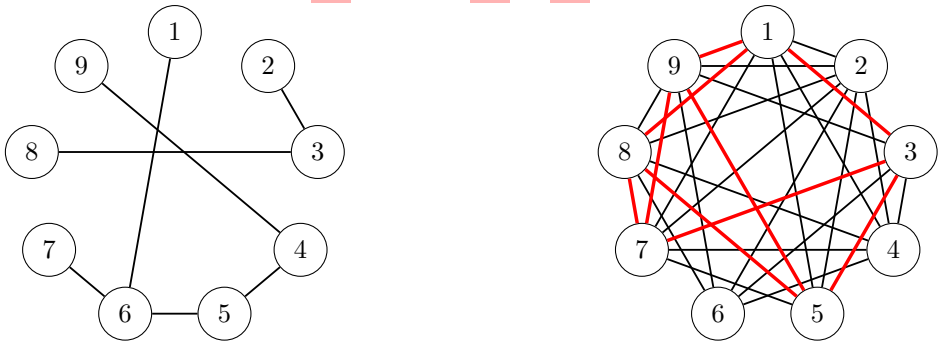


Figure 1 Top: A 9×9 partial matrix M that excludes 4-block pattern. It contains a 3-block pattern formed by rows $\{3, 8, 9\}$ and columns $\{1, 5, 7\}$. Bottom Left: the graph G underlying the partial EDM matrix M. The adjacencies of G are formed by the unspecified entries in the matrix. Bottom Right: the complement graph $\overline{G}$ induced by the unspecified entries of M. The red edges of $\overline{G}$ form $K_{3,3}$, however $\overline{G}$ does not contain $K_{4,4}$.

The algorithm in Theorem 4 compresses the input instance in polynomial time to an equivalent matrix whose size is independent of n. This compression, however, does not guarantee that the numerical entries of the reduced matrix depend only on the parameters t and d; hence it is not a kernel in the strict parameterized complexity sense [17, 25].

Moreover, Theorem 4 by itself does not yield a complete algorithm for solving d-EDMC and we need the following algorithm, which relies on tools from real algebraic geometry.

▶ **Theorem 5.** *There is an algorithm that, given an $n \times n$ partial matrix M and $d \in \mathbb{N}$, runs in time $2^{\mathcal{O}(n^2 \log n)}$ and correctly decides whether (M, d) is a yes-instance of* d-EDMC.

Now, by combining Theorem 4 with Theorem 5, we obtain the following FPT algorithm for d-EDMC with parameters d and t.

▶ **Corollary 6.** *For every $d, t \in \mathbb{N}$, and $n \times n$ partial matrix M excluding a t-block pattern,* d-EDMC *is solvable in time $2^{(d+1)^{\mathcal{O}(t^2)}} + n^{\mathcal{O}(1)}$.*

▶ Remark 7 (Generality of Theorem 4 and Corollary 6). A common way to capture the structure of a partial matrix, see e.g. [2, 7], is to encode it by an underlying graph G as follows. Let G be a graph with vertex set $V(G) = \{1, \ldots, n\}$. An $n \times n$ symmetric hollow partial matrix $M = (m_{ij})$ is said to be a *G-partial* matrix if the entry m_{ij} is specified[2] if and only if $ij \in E(G)$. We refer to G as the *underlying graph* of M, see Figure 1. By studying partial

[2] In this paper, the specified entries of M will always be in $\mathbb{R}_{\geq 0}$.

matrices through a graph-theoretic lens motivated by this defintion, we can identify direct correspondences between their properties and those of their underlying graphs. The condition of Theorem 1 also translates naturally to the property that the average vertex degree of $\overline{G}$ is at most ε. For instance, saying that a partial matrix M excludes a t-block pattern is equivalent to saying that the complement $\overline{G}$ of its underlying graph G does not contain the complete bipartite graph $K_{t,t}$ as a subgraph.

Although Theorem 1 shows that d-EDMC remains intractable even on matrices where the underlying graph's complement has constant average degree, Theorem 4 and Corollary 6 demonstrate that imposing $K_{t,t}$-freeness on the complement of the underlying graph (i.e., excluding a t-block pattern in the input matrix) radically alters the complexity landscape and leads to tractability for a large class of instances.

Many widely studied graph classes with numerous applications can be expressed as $K_{t,t}$-free graphs for a suitable t. In particular, planar graphs are $K_{3,3}$-free by Kuratowski's theorem, graphs excluding a fixed minor H are $K_{t,t}$-free for some t depending on H, graphs of bounded expansion and nowhere dense graphs are $K_{t,t}$-free for an appropriate t, and any graph of maximum degree Δ (or more generally, bounded degeneracy) is $K_{t,t}$-free for $t = \Delta + 1$ or t equal to one plus the degeneracy.

In the rest of the paper, we freely switch between the graph and matrix viewpoints when describing structural properties of partial matrices.

Notice that in the more restricted case when every row of M contains at most Δ unspecified entries, the maximum vertex degree of $\overline{G}$ is at most Δ. In this case, we show that the size of the compression provided by Theorem 4 can be refined as follows.

▶ **Theorem 8.** *There is a polynomial-time algorithm that, given an instance (M, d) of* d-EDMC *such that every row of M has at most Δ unspecified entries, either solves the problem, or outputs an equivalent instance (M', d), such that M' is a principal $n' \times n'$ submatrix of M, where $n' \leq (d+1)(\Delta+1)^2$.*

Combined with Theorem 5, we have the following algorithm.

▶ **Corollary 9.** *For every $d, \Delta \in \mathbb{N}$, and $n \times n$ partial matrix M such that every row of M has at most Δ unspecified entries,* d-EDMC *is solvable in time $2^{\mathcal{O}(d^2 \cdot \Delta^4 \cdot (\log d + \log \Delta))} + n^{\mathcal{O}(1)}$.*

1.1.2 Distance from triviality II

Consider the case where all specified entries of M lie in one fully specified principal submatrix M'. In this case, d-EDMC is polynomial-time solvable: one first determines the minimum dimension d for which M' is isometrically embeddable in $\mathbb{R}^d$ and then assigns values to the remaining entries of M so that the new points lie at appropriate distances. So, a natural definition of distance to this notion of triviality is the least number of fully specified principal submatrices such that their entries cover all specified entries of M. In terms of graphs, if all specified entries of M lie in one fully specified principal submatrix, then the underlying graph G consists of a clique and isolated vertices. Thus all edges of G are covered by one clique. So, when we say that the specified entries in M can be covered by k fully specified principal submatrices, this is equivalent to saying that the underlying graph G admits an *edge clique cover* of size k, that is, the edges of G can be covered by at most k cliques.

▶ Remark 10. A small edge clique cover does *not* imply that $\overline{G}$ is $K_{t,t}$-free for any fixed t. Indeed, the disjoint union of two copies of K_n can be covered by just two cliques, yet its complement contains $K_{n,n}$. Thus, the edge clique cover parameter is *incomparable* with the assumption of $K_{t,t}$-freeness, which motivates the parameterization by edge clique cover.

▶ **Theorem 11.** *There is a polynomial-time algorithm that, given an instance (M, d) of d-EDMC in which M is an $n \times n$ G-partial matrix and the graph G is given together with an edge clique cover $\mathcal{C}$ of size k, either solves the problem, or outputs an equivalent instance (M', d), such that M' is an $n' \times n'$ principal submatrix of M, where $n' \leq (d+1)k^2$.*

▶ **Corollary 12.** *For every $d, k \in \mathbb{N}$, and $n \times n$ partial matrix M such that the specified entries in M can be covered by at most k fully specified principal submatrices, d-EDMC is solvable in time $2^{\mathcal{O}(d^2 \cdot k^4 \cdot (\log d + \log k))} + 2^{2^{\mathcal{O}(k)}} \cdot n^{\mathcal{O}(1)}$.*

Let us next give a brief overview of the proof technique behind Theorem 4, Theorem 8 and Theorem 11. Say that an instance (M, d) of d-EDMC is *efficiently α-reducible* if there is a polynomial-time computable pair comprising a fully specified $\alpha \times \alpha$ principal submatrix indexed by $X \subseteq [n]$ and an element $w \in X$ such that (M, d) is equivalent to $(M - w, d)$. Here, $M - w$ is defined as the matrix obtained from M by removing the row and column indexed by w. The element w is called an *irrelevant* element. Clearly as long as such a pair X and w can be found, we can iteratively reduce the instance by deleting w. To prove these three theorems, we show the following, where G is the underlying graph of M.

- if $\overline{G}$ is $K_{t,t}$-free, then either the instance is solvable in polynomial time or it is already compressed to the claimed size or it is efficiently $d^{\mathcal{O}(t)}$-reducible; and
- if $\overline{G}$ has max-degree at most Δ, then either the instance is solvable in polynomial time or it is already compressed to the claimed size or it is efficiently $\mathcal{O}(d \cdot \Delta)$-reducible; and
- given an edge clique cover of size k for G, either the instance is solvable in polynomial time or it is already compressed to the claimed size or it is efficiently $\mathcal{O}(d \cdot k)$-reducible.

Proving each of the above statements is done along similar lines, but there are certain instance-specific aspects that we exploit in each case. In fact, for the first two statements, we show the existence of an α depending only on the parameters such that the index set of *any* fully specified $\alpha \times \alpha$ principal submatrix must contain an irrelevant element. Then, we can use the sparsity of $\overline{G}$ to infer the existence of such a fully specified principal submatrix of M. For the third statement, we observe that one of the cliques in the given edge clique cover of G must have size at least α (where α depends on the parameters) and then we argue that such a clique contains an irrelevant element.

1.1.3 Distance from triviality III

Our final result concerns a further class of matrices that admit polynomial-time completion algorithms. Recall that a graph G is *chordal* if it has no induced cycles of length greater than 3, see Figure 2. Bakonyi and Johnson [4] proved the following: when G is chordal, a G-partial matrix M can be completed to an EDM that is realizable in $\mathbb{R}^d$ if and only if the submatrix induced by the vertex set of each maximal clique of G is itself an EDM realizable in $\mathbb{R}^d$. Since an n-vertex chordal graph has at most n maximal cliques and these can be listed in polynomial time [27], Laurent [33] observed that this characterization yields a straightforward polynomial-time algorithm for d-EDMC as follows. For each maximal clique of G one checks, in polynomial time, whether the corresponding principal submatrix of M (all of whose entries are specified, by definition) is an EDM realizable in $\mathbb{R}^d$. So, considering partial matrices with chordal underlying graphs as our notion of triviality, let us address the next notion of distance from triviality.

The popular measure of the distance of a graph G to a chordal graph is the *fill-in* of a graph, which is the minimum number of edges that should be added to G to make it chordal. Our next theorem extends the polynomial-time algorithm of Bakonyi and Johnson [4] for chordal graphs to graphs with constant fill-in.

$$\begin{bmatrix} 0 & 1 & 1 & 1 & * & * & * & * & * \\ 1 & 0 & 1 & 1 & * & * & * & * & * \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & * & * \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & * & * \\ * & * & 1 & 1 & 0 & 1 & 1 & 1 & * \\ * & * & 1 & 1 & 1 & 0 & 1 & * & 1 \\ * & * & 1 & 1 & 1 & 1 & 0 & 1 & * \\ * & * & * & * & 1 & * & 1 & 0 & 1 \\ * & * & * & * & * & 1 & * & 1 & 0 \end{bmatrix}$$

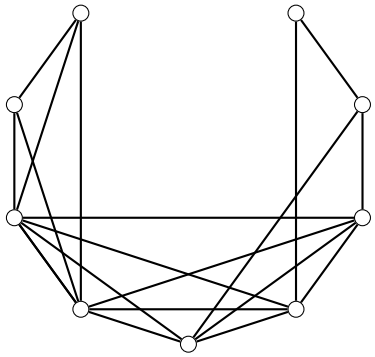


Figure 2 A partial EDM matrix whose underlying graph is chordal.

▶ **Theorem 13.** *Let M be an $n \times n$ G-partial matrix and let $d \in \mathbb{N}$. There is an algorithm that runs in $d^{\mathcal{O}(kd)}\ 2^{\mathcal{O}(k^2)}\ n^{\mathcal{O}(k)}$ time, where k is the size of a minimum fill-in of G, and correctly decides whether (M, d) is a yes-instance of d-EDMC.*

Laurent [33] presented an XP algorithm, that is, polynomial for every fixed k, which decides whether a G-partial matrix can be completed to an EDM realizable in *some* Euclidean space, where k is the minimum size of a fill-in of G. This algorithm, however, does not determine the *smallest* dimension d that suffices, and therefore does not solve d-EDMC. Theorem 13 can thus be viewed as a non-trivial extension of Laurent's result, tailored specifically for embeddability into a prescribed dimension d.

We next give a brief outline of the techniques behind this algorithm. Due to the result of Bakonyi and Johnson [4], it is necessary and sufficient to decide whether a given G-partial matrix M can be extended to a G'-partial matrix M' satisfying the following conditions: (i) G' is chordal, (ii) G' is a supergraph of G, and (iii) for every maximal clique of G', the submatrix of M' induced by the clique's vertex set is an EDM realizable in $\mathbb{R}^d$. For us, G' is defined by a fill-in set X, that is, $G' = G + X$, where X is a set of non-edges of G whose addition to G makes it chordal. Thus, the task reduces to computing M' by assigning values to the entries corresponding to the edges in X, ensuring that condition (iii) is satisfied. To do so, we reduce this task to testing the truth of an existentially quantified statement over a bounded set of polynomial equations and inequalities and then finally invoke an algorithm of Basu, Pollack, and Roy [5]. We note that a similar approach was used in [6], however our case is more complex as the maximal cliques of G' may share edges, requiring us to handle multiple EDMs simultaneously in this reduction.

1.2 Related work

The study of *Euclidean distance matrices* (EDMs) dates back to the pioneering works of Cayley [13] and Menger [34]. Foundational results were later obtained by Schoenberg [36] and by Young and Householder [41]. The subject was further developed in a series of papers by Gower, Critchley, Farebrother, and others [28, 29, 16, 24]. Schoenberg [36] also established the deep connection between EDMs and positive-semidefinite (PSD) matrices. A comprehensive treatment of the vast literature on EDMs lies beyond the scope of this article; we refer the interested reader to the monographs and surveys [10, 15, 19, 32, 18, 2].

2 Preliminaries

For every $x \in \mathbb{N}_{>0}$, we use $[x]$ to denote the set $\{1, \ldots, x\}$.

Distance spaces and matrices. Let X be a set. A function $\rho\colon X \times X \to \mathbb{R}_{\geq 0}$ is a *distance* on X if: (i) ρ is symmetric, that is, for any $x, y \in X$, $\rho(x, y) = \rho(y, x)$, and (ii) $\rho(x, x) = 0$ for all $x \in X$. Then, (X, ρ) is called a *distance space*. A distance space (X, ρ) for a finite $X = \{x_1, \ldots, x_n\}$ can be equivalently defined by the *distance matrix*, in which the value in row i and column j is ρ_{ij}^2, where $\rho_{ij} = \rho(x_i, x_j)$ for $i, j \in \{1, \ldots, n\}$. Throughout the paper we do not distinguish metric spaces and the corresponding distance matrices.

A distance space (X, ρ) is *isometrically embeddable* into $\mathbb{R}^d$ if there is a map, called *isometric embedding*, $\varphi\colon X \to \mathbb{R}^d$ such that $\rho(x, y) = \|\varphi(x) - \varphi(y)\|_2$ for all $x, y \in X$, where $\|p - q\|_2 = \sqrt{\langle p, p\rangle + \langle q, q\rangle - 2\langle p, q\rangle}$ for $p, q \in \mathbb{R}^d$. Notice that we do not require φ to be injective, that is, several points of (X, ρ) may be mapped to the same point of $\mathbb{R}^d$. Throughout the paper, whenever we mention an embedding, we mean an isometric embedding. Moreover, when we use the term d-embedding, we are referring to embedding into $\mathbb{R}^d$. A d-embeddable distance space is *strongly d-embeddable* if it is not $(d-1)$-embeddable. As convention, we assume that the empty set of points is d-embeddable for every d. A symmetric $n \times n$ matrix $D = (d_{ij})$ over $\mathbb{R}_{\geq 0}$ with $d_{ii} = 0$ for all $i \in \{1, \ldots, n\}$ is a *Euclidean distance matrix* (EDM) in $\mathbb{R}^d$ if $D = D(\rho)$ is the distance matrix of a distance space (X, ρ) embeddable in $\mathbb{R}^d$. Suppose that $\mathcal{X} = (X, \rho)$ is embeddable into $\mathbb{R}^d$. The ordered set $P = (p_1, \ldots, p_n)$ of points in $\mathbb{R}^d$ is said to be a *realization* of $\mathcal{X}$ if there is an embedding $\varphi\colon X \to \mathbb{R}^d$ such that $\varphi(x_i) = p_i$ for all $i \in \{1, \ldots, n\}$.

For a d-embeddable distance space (X, ρ), a set $Y \subseteq X$ is a *metric basis* if, given an isometric embedding φ of (Y, ρ) into $\mathbb{R}^d$, there is a unique way to extend φ to an isometric embedding of (X, ρ). Equivalently, if a realization of (Y, ρ) is fixed then the embedding of any point of $X \setminus Y$ in a d-embedding of (X, ρ) is unique. In our paper we also use the embeddablity characterization based on the properties of Cayley-Menger matrices. Towards this, we use terminology and notation from [6]. For $r+1$ points $x_0, x_1, \ldots, x_r$ of distance space (X, ρ) the *Cayley-Menger determinant* $CM(x_0, x_1, \ldots, x_r)$ is the determinant of the matrix obtained from the distance matrix induced by these points by prepending a row and a column whose first element is zero and the other elements are one.

▶ **Proposition 14** ([10, Chapter IV]). *A distance space $\mathcal{X} = (X, \rho)$ with n points is strongly d-embeddable if and only if there exist $d+1$ points, say $X_d = \{x_0, \ldots, x_d\}$, such that:*

1. $(-1)^{j+1} CM(x_0, x_1, \ldots, x_j) > 0$ *for* $1 \leq j \leq d$, *and*
2. *for any* $x, y \in X \setminus X_d$,

$$CM(x_0, x_1, \ldots, x_d, x) = CM(x_0, x_1, \ldots, x_d, y) = CM(x_0, x_1, \ldots, x_d, x, y) = 0.$$

Equivalently (see, for example, [37]), $\mathcal{X}$ is strongly d-embeddable if and only if there is a set of $d+1$ points $X_d = \{x_0, \ldots, x_d\}$ such that $(\{x_0, \ldots, x_j\}, \rho)$ is strongly j-embeddable for all $j \in \{1, \ldots, d\}$, and for every $x, y \in X \setminus X_d$, $(X_d \cup \{x\} \cup \{y\})$ is d-embeddable.

▶ **Proposition 15** ([10, Chapter IV]). *Let (X, ρ) be a d-embeddable distance space, and let $B \subseteq X$ be a metric basis. Then an embedding of B in $\mathbb{R}^d$ is unique up to rigid transformations, and given an embedding φ of (B, ρ) into $\mathbb{R}^d$, there is a unique way to extend φ to an embedding of (X, ρ). Equivalently, if a realization of (B, ρ) is fixed then the embedding of any point of $X \setminus B$ in a d-embedding of (X, ρ) is unique.*

Let (X, ρ) be a distance space. For a nonnegative integer r, we say that $Y \subseteq X$ of size $r+1$ is *independent* if (Y, ρ) is strongly r-embeddable.

▶ **Proposition 16** ([10, Chapter IV]). *Let (X, ρ) be a strongly d-embeddable distance space. Then the following hold:* (i) *any single-element set is independent,* (ii) *if $Y \subseteq X$ is independent, then any $\emptyset \subset Z \subseteq Y$ is independent, and* (iii) *if $Y, Z \subseteq X$ are independent and $|Y| > |Z|$ then there is a $y \in Y \setminus Z$ such that $Z \cup \{y\}$ is independent,* (iv) *the maximum size of an independent set is $d+1$ and any independent set Y of size $d+1$ is a metric basis.*

▶ **Proposition 17** ([10, Chapter IV] and [1, 38]). *Given a distance space $\mathcal{X} = (X, \rho)$ with n points and a positive integer d, in $\mathcal{O}(n^3)$ time, it can be decided whether $\mathcal{X}$ can be embedded into $\mathbb{R}^d$ and, if such an embedding exists, then a metric basis and a realization can be constructed in this running time.*

We remind that by our convention, we do not distinguish metric spaces and the corresponding distance matrices. Thus, we say that an n-tuple of points of $\mathbb{R}^d$ is a realization of an $n \times n$ distance matrix. Furthermore, we say that a subset of indices $X \subseteq \{1, \ldots, n\}$ is a metric basis of the distance matrix meaning that the corresponding points of the distance space compose a metric basis for it. In this spirit, if M is the distance matrix of a (strongly) d-embeddable distance space, then we say that M is a *(strongly) d-embeddable EDM*.

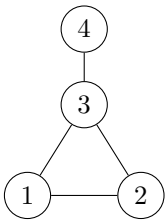
Graphs and matrices. We consider simple finite undirected graphs and refer to [20] for the standard graph-theoretic notation. Given a graph G, we denote by $V(G)$ and $E(G)$ the sets of vertices and edges, respectively. Throughout the paper we use n and m for $|V(G)|$ and $|E(G)|$, respectively, if the graph is clear from the context. A set of pairwise adjacent vertices K of G is called a *clique* and a set of pairwise non-adjacent vertices is *independent*. A family of k cliques $\mathcal{C} = \{C_1, \ldots, C_k\}$ of a graph G is en *edge clique cover* of G if for every edge $uv \in E(G)$, there is $i \in \{1, \ldots, k\}$ such that uv is *covered* by C_i, that is, $u, v \in C_i$. For a vertex $v \in V(G)$, we use $N_G(v)$ to denote the *open neighborhood* of v, that is, the set of adjacent to v vertices. The *degree* $d_G(v) = |N_G(v)|$, and the *maximum degree* of G is $\Delta(G) = \max\{d_G(v) \mid v \in V(G)\}$. For a graph G, its *complement* $\overline{G}$ is the graph with the same set of vertices as G, and two distinct vertices $u, v \in V(G)$ are adjacent to $\overline{G}$ if and only if they are not adjacent in G.

Let A be an $n \times n$ matrix. The matrix obtained from A by deleting $n-k$ rows and $n-k'$ columns (with $1 \leq k, k' \leq n$) is called a $k \times k'$ *submatrix* of A. A *principal submatrix* of A is a square submatrix obtained by deleting the same set of row and column indices; that is, if the i-th row of A is deleted, then the i-th column is also deleted. Let G be a graph with $V(G) = \{1, \ldots, n\}$. An $n \times n$ matrix $A = (a_{ij})$ over $\mathbb{R}_{\geq 0}$ with some unspecified elements is said to be a *G-partial* matrix if (i) the entry a_{ij} is defined (or specified) if and only if $ij \in E(G)$, (ii) $a_{ij} = a_{ji}$ for all $ij \in E(G)$, and $a_{ii} = 0$ for all $i \in \{1, \ldots, n\}$. We also say that G is the *underlying* graph of A. Let A be a G-partial matrix and let d be a positive integer. A matrix D is said to be a d-EDM *completion* of A if:

1. D is d-embeddable, and
2. $d_{ij} = a_{ij}$ for all $ij \in E(G)$.

See Figure 3 for an example.

It is convenient for us to combine graph-theoretic and matrix notation. Let $A = (a_{ij})$ be symmetric $n \times n$ matrix and let G be a graph with $V(G) = [n]$. For a set $X \subseteq [n]$, we use $G[X]$ to denote the subgraph of G induced by X and we use $A[X]$ to denote the principal submatrix indexed by X, that is, the submatrix of A composed by the elements a_{ij} with $i, j \in X$. We also write $G - X$ and $A - X$ to denote the graph obtained by the deletion



$$A = \begin{bmatrix} 0 & 1 & 1.25 & * \\ 1 & 0 & 1.25 & * \\ 1.25 & 1.25 & 0 & 1 \\ * & * & 1 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 1 & 1.25 & 4.25 \\ 1 & 0 & 1.25 & 4.25 \\ 1.25 & 1.25 & 0 & 1 \\ 4.25 & 4.25 & 1 & 0 \end{bmatrix}$$

Figure 3 Graph G, G-partial matrix A, and EDM completion D of A. The distance space $\mathcal{X}$ defined by D is embeddable into $\mathbb{R}^2$. For example, a realization φ of $\mathcal{X}$ would map the elements of $\mathcal{X}$ to points $p_1 = (0,0)$, $p_2 = (1,0)$, $p_3 = (0.5,1)$, and $p_4 = (0.5,2)$.

of the vertices of X and the submatrix obtained by the deletion of the rows and columns indexed by X, respectively, that is, $G - X = G[V(G) \setminus X]$ and $A - X = A[V(G) \setminus X]$. For a set $X = \{x\}$, we write $G - x$ and $A - x$ instead of $G - \{x\}$ and $A - \{x\}$, respectively.

We conclude this section with Theorem 1, which shows that for tractability of d-EDMC, it is not sufficient to require high density of the specified element in the input matrix (or, equivalently, high density of the underlying graph G).

▶ **Theorem 1.** *$\forall \varepsilon \in (0,1)$, d-EDMC remains strongly NP-hard even for instances (M,d) with $d = 2$ in which the $n \times n$ matrix M contains at most εn unspecified entries.*

3 Compressing G-partial matrices when G is dense (distance from triviality I and II)

This section is devoted to proving Theorem 4, Theorem 8 and Theorem 11.

Proof of Theorem 4. The main step of the proof is to identify an "irrelevant" vertex of G in polynomial time, that is, a vertex whose deletion does not transform a no-instance into a yes-instance. By exhaustively repeating this step of removing an irrelevant vertex we either solve the problem, or construct an equivalent instance on $(d+1)^{\mathcal{O}(t^2)}$ rows and columns.

Let M be a G-partial matrix, where G excludes a t-block pattern. If $t = 1$, then M is fully specified (i.e., G is complete) and so, we solve the instance (M,d) using Proposition 17. So, we assume that $t > 1$. Let $\eta(d,t) = (d+1)t + (t-1)(d+1)^{t+1} + 1$, and $\rho(d,t) = \binom{2t+\eta(d,t)-2}{2t-1}$.

If $n < \rho(d,t)$, then the desired compression is already achieved and we simply output (M,d). From now on, we assume that $n \geq \rho(d,t)$. Since $\overline{G}$ is $K_{t,t}$-free, it does not contain a clique of size $2t$. By Ramsey's classic theorem [22] $\overline{G}$ must therefore contain an independent set X of size $\eta(d,t)$. Moreover, such a set X can be computed in polynomial time. By definition, X induces a clique in G. Notice that because M is not fully specified, $V(G) \neq X$.

We use Proposition 17 to check in polynomial time whether $M[X]$ is d-embeddable. If not, then we report that (M,d) is a no-instance and stop. For every $v \in V(G) \setminus X$, notice that $C_v = \{v\} \cup (N_G(v) \cap X)$ is a clique of G. So, for every $v \in V(G) \setminus X$ we also check whether $M[C_v]$ is d-embeddable, and report that (M,d) is a no-instance if it is not.

We next iteratively construct disjoint sets $X_1, \ldots, X_t \subseteq X$ of size at most $d+1$ as follows. Assuming that $X_0 = \emptyset$, we select X_i to be a metric basis of $M[X \setminus \bigcup_{j=0}^{i-1} X_j]$ for $i \in \{1, \ldots, t\}$ obtained using Proposition 17. As $|X_i| \leq d+1$ for $i \in \{1, \ldots, t\}$, we have that $|\bigcup_{i=1}^{t} X_i| \leq (d+1)t$. Moreover, since X is large enough, the sets $X_1, \ldots, X_t$ exist.

Consider $Y = \{y \in V(G) \setminus X \colon X_i \setminus N_G(y) \neq \emptyset \text{ for all } i \in \{1, \ldots, t\}\}$. That is, Y consists of all vertices outside X having a non-neighbor in each X_i in G.

▷ Claim 18. $|Y| \leq (t-1)(d+1)^t$.

For every $y \in Y$, we consider the clique $K_y = N_G(y) \cap X$. Since $K_y \subseteq C_y$ and we have already concluded that $M[C_y]$ is d-embeddable, we have that $M[K_y]$ is d-embeddable. We compute a metric basis Z_y of $M[K_y]$ using Proposition 17. Because $|Y| \leq (t-1)(d+1)^t$ by Claim 18 and $|Z_y| \leq d+1$ for $y \in Y$, we have that $|\bigcup_{y \in Y} Z_y| \leq (t-1)(d+1)^{t+1}$. Because $|\bigcup_{i=1}^{t} X_i| \leq (d+1)t$ and $|X| \geq (d+1)t + (t-1)(d+1)^{t+1} + 1$, we have that there is $w \in X \setminus \left(\left(\bigcup_{i=1}^{t} X_i\right) \cup \left(\bigcup_{y \in Y} Z_y\right)\right)$.

▷ Claim 19. The instances (M, d) and $(M - w, d)$ of d-EDMC are equivalent.

We set $M := M - w$ and iterate. In at most n rounds, we either solve the problem or obtain an equivalent instance (M', d), where M' is a $n' \times n'$ submatrix of M with $n' < \binom{2t+\eta(d,t)-2}{2t-1}$. For the running time, note that the construction of the independent set X as well as that of $X_1, \ldots, X_t$ is done in polynomial time by Proposition 17. Then Y and the sets Z_y for $y \in Y$ can also be constructed in polynomial time by Proposition 17. So, an irrelevant vertex can be found in polynomial time. Because the total number of rounds is at most n, the overall running time is polynomial. ◀

As discussed, by combining Theorem 4 with Theorem 5, we obtain the following corollary.

▶ **Corollary 6.** *For every $d, t \in \mathbb{N}$, and $n \times n$ partial matrix M excluding a t-block pattern, d-EDMC is solvable in time $2^{(d+1)^{\mathcal{O}(t^2)}} + n^{\mathcal{O}(1)}$.*

In the remaining part of the section, we show that for some other classes of dense graphs G, we can obtain a better compression for d-EDMC than the one obtained by using Theorem 4 as a black box. The proofs of these results still follow the same lines as the proof of Theorem 4. First, we consider the case when $\overline{G}$ has bounded maximum degree. Since a graph of maximum degree at most Δ does not contain $K_{\Delta+1,\Delta+1}$, in the following theorem, we consider a subclass of matrices considered in Theorem 4.

▶ **Theorem 8.** *There is a polynomial-time algorithm that, given an instance (M, d) of d-EDMC such that every row of M has at most Δ unspecified entries, either solves the problem, or outputs an equivalent instance (M', d), such that M' is a principal $n' \times n'$ submatrix of M, where $n' \leq (d+1)(\Delta+1)^2$.*

Finally in this section, we prove Theorem 11. Notice that Theorem 11 assumes that an edge clique cover is given. However, an edge clique cover of size at most k can be found in FPT in k time (if exists) by the results of Gramm et al. [30].

▶ **Proposition 20** ([30])**.** *Given a graph G and a positive integer k, it can be decided in $2^{2^{\mathcal{O}(k)}} n^{\mathcal{O}(1)}$ time whether G admits an edge clique cover of size at most k. Furthermore, an edge clique cover can be found in the same time if it exists.*

Then we obtain the following corollary.

▶ **Corollary 12.** *For every $d, k \in \mathbb{N}$, and $n \times n$ partial matrix M such that the specified entries in M can be covered by at most k fully specified principal submatrices, d-EDMC is solvable in time $2^{\mathcal{O}(d^2 \cdot k^4 \cdot (\log d + \log k))} + 2^{2^{\mathcal{O}(k)}} \cdot n^{\mathcal{O}(1)}$.*

4 The case of bounded fill-in (distance from triviality III)

In this section, we present our algorithm for d-EDMC for G-partial matrices when G is almost chordal, i.e., it has a small fill-in. Recall that the minimum fill-in of a graph G is the size of a smallest set of non-edges whose addition to G makes it chordal.

▶ **Theorem 13.** *Let M be an $n \times n$ G-partial matrix and let $d \in \mathbb{N}$. There is an algorithm that runs in $d^{\mathcal{O}(kd)}\ 2^{\mathcal{O}(k^2)}\ n^{\mathcal{O}(k)}$ time, where k is the size of a minimum fill-in of G, and correctly decides whether (M, d) is a yes-instance of d-EDMC.*

Towards the proof of this theorem, let us prepare as follows.

▶ **Proposition 21** ([4]). *Every G-partial matrix M where G is chordal can be completed to an EDM in $\mathbb{R}^d$ if every fully specified principal submatrix of M is d-embeddable.*

▶ **Definition 22** ([5]). Let R be a real closed field and $\mathcal{P} \subset R[X_1, \ldots, X_t]$ be a finite set of polynomials. A *$\mathcal{P}$-atom* is one of $P = 0, P \neq 0$, $P \geq 0$, $P \leq 0$, where P is a polynomial in $\mathcal{P}$ and a *quantifier-free $\mathcal{P}$-formula* is a formula constructed from $\mathcal{P}$-atoms together with the logical connectives $\wedge$, $\vee$ and $\neg$. The *R-realization* of a formula F with free variables $x_1, \ldots, x_r$ is a mapping $\nu : \{x_1, \ldots, x_r\} \to R$ so that the sentence resulting from F (denoted by $F[\nu]$) by instantiating each free variable x with $\nu(x)$ is true.

▶ **Proposition 23** (Theorem 13.13, [5]). *Let $s, \ell \in \mathbb{N}$. Let $(\exists X_1) \ldots (\exists X_t) F(X_1, \ldots, X_t)$, be a sentence, where $F(X_1, \ldots, X_t)$ is a quantifier free $\mathcal{P}$-formula where $\mathcal{P} \subset R[X_1, \ldots, X_t]$ is a set of at most s polynomials each of degree at most ℓ. There exists an algorithm to decide the truth of the sentence with complexity[3] $s^{t+1} \cdot \ell^{\mathcal{O}(t)}$ in D where D is the ring generated by the coefficients of the polynomials in $\mathcal{P}_\ell$.*

▶ **Definition 24.** Let M be a G-partial matrix where $V(G) = [n]$ and let Z denote the set of pairs in $\binom{[n]}{2}$ such that every pair in Z indexes a pair of unspecified entries of M, that is, for every $\{i, j\} \in Z$, M_{ij} and M_{ji} are both unspecified. Define the matrix $\hat{M}$ as follows. If $\{i, j\} \notin Z$, then define $\hat{M}_{ij} = M_{ij}$, otherwise define $\hat{M}_{ij} = z_{ij}$ where z_{ij} is an indeterminate.

We have the following adaptation of the notion of Z-augmented Cayley-Menger determinant from [6].

▶ **Definition 25.** Let M and Z be as described in Definition 24. Let $I = \{x_0, \ldots, x_r\} \subseteq [n]$. The *$Z$-Augmented Cayley-Menger determinant indexed by I* is obtained from the Cayley-Menger determinant by replacing each $\rho^2_{x_i, x_j}$ with $\hat{M}_{x_i, x_j}$.

▶ **Lemma 26.** *Let M, Z and $I = \{x_0, \ldots, x_r\}$ be as described in Definition 25. Then, $CM_Z(x_0, x_1, \ldots, x_r)$ is a multi-variate polynomial with real coefficients, over the set $\{z_{ij} \mid \{i, j\} \in Z\}$ of indeterminates where each monomial has degree at most $\min(|I|, 2|Z|)$.*

▶ **Lemma 27.** *There is an algorithm that, given M, I, Z as described in Definition 25, runs in time $|I|^{\mathcal{O}(|I|)} \cdot n^{\mathcal{O}(1)}$ and produces the polynomial representing the Z-Augmented Cayley-Menger determinant indexed by I.*

▶ **Lemma 28.** *Consider an $n \times n$ EDM M and $Y \subseteq [n]$. Let $B \subseteq [n]$ be a metric basis of $M - Y$. Then, there is a metric basis of M that contains B.*

[3] The measure of complexity here is the number of arithmetic operations. Recall that we use the real RAM model.

Proof of Theorem 13. Let the instance (M, d) be given, where M is G-partial and moreover, let X be a set of at most k non-edges of G such that $G' = G + X$ is a chordal graph. It is straightforward to construct G from M and once we have G, we compute X (if it exists) in time $2^{o(k)}n^{\mathcal{O}(1)}$ using the algorithm of Fomin and Villanger [26]. In the rest of the proof, we will only refer to the graph G', so we can refer to the pairs in X as edges. Let $\mathcal{Z}$ denote a set of $2k$ indeterminates, two per edge in X. Formally, for every edge $uv \in X$, $\mathcal{Z}$ contains the indeterminates z_{uv} and z_{vu}. Let $C_1, \ldots, C_t$ be the maximal cliques in G'. It is well-known that $t \leq n$ and these cliques can be computed in polynomial time [27].

1. Do the following for each $i \in [t]$:
   - **a.** Define $\mathcal{Z}_i = \{z_{uv} \mid u, v \in V(C_i)\}$. That is, $\mathcal{Z}_i$ comprises those indeterminates corresponding to those edges in X that have both endpoints in $V(C_i)$.
   - **b.** Define $V_X(C_i) = \{u \mid \exists v : uv \in X \wedge \{u, v\} \subseteq V(C_i)\}$. That is, $V_X(C_i)$ is the set of endpoints of edges in X that are contained in C_i.
   - **c.** Define B_i to be an arbitrary metric basis of $M_i = M[V(C_i) \setminus V_X(C_i)]$. Since by definition, M_i is fully specified, B_i is well-defined. If $|B_i| > d + 1$, then stop and conclude that the input is a no-instance.
   - **d.** For every $Y \subseteq V_X(C_i)$ of size at most $d + 1 - |B_i|$, do the following:
     - **i.** Define $B_i^Y = B_i \cup Y = \{x_0, \ldots, x_r\}$.
     - **ii.** Use Lemma 27 to construct the following set P_i^Y of atoms:
       - **A.** $(-1)^{j+1} CM_{Z_i}(x_0, x_1, \ldots, x_j) > 0$, where $1 \leq j \leq r$.
       - **B.** $CM_{Z_i}(x_0, x_1, \ldots, x_r, x) = 0$ for every $x \in V(C_i) \setminus B_i^Y$.
       - **C.** $CM_{Z_i}(x_0, x_1, \ldots, x_r, x, y) = 0$ for every $x, y \in V(C_i) \setminus B_i^Y$.
       - **D.** $z_{uv} \geq 0$ for each indeterminate $z_{uv} \in Z_i$.
       - **E.** $z_{uv} - z_{vu} = 0$ for each pair z_{uv}, z_{vu} of indeterminates in Z_i.
     - **iii.** Define ϕ_i^Y to be the quantifier-free formula defined as the conjunction of the atoms in P_i^Y.
   - **e.** Define Ψ_i to be the quantifier-free formula defined as the disjunction of the formulas ϕ_i^Y taken over all $Y \subseteq V_X(C_i)$ of size at most $d + 1 - |B_i|$.
2. Define the quantifier-free formula $\Gamma = \bigwedge_{i \in [t]} \Psi_i$.
3. Run the algorithm of Proposition 23 on the sentence obtained by existentially quantifying the indeterminates in $\mathcal{Z}$ and prepending them to Γ. Return the same answer as that produced by this invocation.

▷ Claim 29. (M, d) is a yes-instance of d-EDMC if and only if this algorithm returns Yes.

▷ Claim 30. For each $i \in [t]$ and $Y \subseteq V_X(C_i)$, the atoms in P_i^Y are $\mathcal{P}_i^Y$-atoms, where $\mathcal{P}_i^Y$ is a set of $\mathcal{O}(n^2)$ polynomials each of degree at most $\min(d + 3, 2k)$.

▷ Claim 31. The algorithm runs in time $d^{\mathcal{O}(kd)} \cdot 2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(k)}$.

These three claims complete the proof of the theorem. ◀

5 Concluding remarks and open questions

By Theorem 13, the d-EDMC problem can be solved in polynomial time for every fixed value of d and k. Moreover, for fixed values of k, our algorithm is FPT in d. It is therefore a natural question whether the problem is in FPT with respect to *both* d and k.

References

1 Jorge Alencar, Tibérius O. Bonates, Carlile Lavor, and Leo Liberti. An algorithm for realizing euclidean distance matrices. *Electron. Notes Discret. Math.*, 50:397–402, 2015. doi:10.1016/J.ENDM.2015.07.066.

2 Abdo Y. Alfakih. *Euclidean distance matrices and their applications in rigidity theory*. Springer, 2018.

3 A.Y. Alfakih, A. Khandani, and H. Wolkowicz. Solving euclidean distance matrix completion problems via semidefinite programming. *Comput. Optim. Appl.*, 12:13–30, 1999. doi:10.1023/A:1008655427845.

4 M. Bakonyi and C. Johnson. The euclidean distance matrix completion problem. *SIAM J. Matrix Anal. Appl.*, 16:646–654, 1995. doi:10.1137/S0895479893249757.

5 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

6 Matthias Bentert, Fedor V. Fomin, Petr A. Golovach, M. S. Ramanujan, and Saket Saurabh. When distances lie: Euclidean embeddings in the presence of outliers and distance violations. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry, SoCG 2025, June 23-27, 2025, Kanazawa, Japan*, volume 332 of *LIPIcs*, pages 15:1–15:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SOCG.2025.15.

7 Bonnie Berger, Jon M. Kleinberg, and Frank Thomson Leighton. Reconstructing a three-dimensional model with arbitrary errors. *J. ACM*, 46(2):212–235, 1999. doi:10.1145/301970.301972.

8 P. Biswas, T.-C. Liang, K.-C. Toh, T.-C. Wang, and Y. Ye. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE Trans. Autom. Sci. Eng.*, 3:360–371, 2006. doi:10.1109/TASE.2006.877401.

9 P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *3rd International Symposium on Information Processing in Sensor Networks*, pages 46–54, Berkeley, CA, 2004. ACM. doi:10.1145/984622.984630.

10 L.M. Blumenthal. *Theory and Applications of Distance Geometry*. Chelsea, New York, 2nd edition, 1970.

11 E.J. Candes and Y. Plan. Matrix completion with noise. *Proc. IEEE*, 98:925–936, 2010. doi:10.1109/JPROC.2009.2035722.

12 Augustin-Louis Cauchy. Sur les polygones et polyèdres. *Journal de l'École Polytechnique*, 19:87–90, 1813.

13 Arthur Cayley. On the theory of determinants. *Philosophical Magazine*, 19:1–16, 1841.

14 G.M. Crippen. Chemical distance geometry: Current realization and future projection. *J. Math. Chem.*, 6:307–324, 1991.

15 G.M. Crippen and T.F. Havel. *Distance Geometry and Molecular Conformation*, volume 15 of *Chemometrics Series*. Research Studies Press, Taunton, 1988.

16 F. Critchley. On certain linear mappings between inner-product and squared distance matrices. *Linear Algebra Appl.*, 105:91–107, 1988.

17 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

18 J. Dattorro. *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing USA, 2008.

19 Michel Deza, Monique Laurent, and Robert Weismantel. *Geometry of cuts and metrics*, volume 2. Springer, 1997.

20 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 3rd edition, 2005.

21 L. Doherty, K.S.J. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *IEEE INFOCOM*, volume 3, pages 1655–1663, Anchorage, AK, 2001.

22 P. Erdös and G. Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935. URL: http://www.numdam.org/item?id=CM_1935__2__463_0.

23 Haw-ren Fang and Dianne P O'Leary. Euclidean distance matrix completion problems. *Optimization Methods and Software*, 27(4-5):695–717, 2012. doi:10.1080/10556788.2011.643888.

24 R.W. Farebrother. Three theorems with applications to euclidean distance matrices. *Linear Algebra Appl.*, 95:11–16, 1987.

25 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization. Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.

26 Fedor V. Fomin and Yngve Villanger. Subexponential parameterized algorithm for minimum fill-in. *SIAM J. Computing*, 42(6):2197–2216, 2013. doi:10.1137/11085390X.

27 Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

28 J.C. Gower. Euclidean distance geometry. *Math. Sci.*, 7:1–14, 1982.

29 J.C. Gower. Properties of euclidean and non-euclidean distance matrices. *Linear Algebra Appl.*, 67:81–97, 1985.

30 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *ACM Journal of Experimental Algorithmics*, 13, 2008. doi:10.1145/1412228.1412236.

31 T.F. Havel, I.D. Kuntz, and G.M. Crippen. The theory and practice of distance geometry. *Bull. Math. Biol.*, 45:665–720, 1983.

32 M. Laurent. A tour d'horizon on positive semidefinite and euclidean distance matrix completion problems. In *Semidefinite Programming and Interior-Point Approaches for Combinatorial Optimization Problems*, Fields Institute Communications, pages 51–76. Amer. Math. Soc., Providence, 1998.

33 Monique Laurent. Polynomial instances of the positive semidefinite and euclidean distance matrix completion problems. *SIAM J. Matrix Anal. Appl.*, 22:874–894, 2001. doi:10.1137/S0895479899352689.

34 Karl Menger. Untersuchungen über allgemeine metrik. *Mathematische Annalen*, 100:75–163, 1928.

35 James B Saxe. Embeddability of weighted graphs in k-space is strongly np-hard. In *17th Allerton Conf. Commun. Control Comput., 1979*, pages 480–489, 1979.

36 Isaac J. Schoenberg. Remarks to Maurice Fréchet's article "Sur la définition axiomatique d'une classe d'espace distanciés vectoriellement applicable sur l'espace de Hilbert". *Annals of Mathematics*, 36:724–732, 1935.

37 Anastasios Sidiropoulos, Dingkang Wang, and Yusu Wang. Metric embeddings with outliers. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 670–689. SIAM, 2017. doi:10.1137/1.9781611974782.43.

38 Manfred J. Sippl and Harold A. Scheraga. Solution of the embedding problem and decomposition of symmetric matrices. *Proc. Nat. Acad. Sci. U.S.A.*, 82(8):2197–2201, 1985. doi:10.1073/pnas.82.8.2197.

39 K.Q. Weinberger, F. Sha, and L.K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *21st International Conference on Machine Learning (ICML)*, page 106, Banff, AB, 2004. ACM.

40 Y. Yemini. Some theoretical aspects of position-location problems. In *20th Annual Symposium on Foundations of Computer Science (SFCS)*, pages 1–8. IEEE, 1979.

41 G. Young and A. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–22, 1938.

Separators for Intersection Graphs of Spheres

Jacob Fox
Department of Mathematics, Stanford University, Stanford, CA, USA

Jonathan Tidor
Department of Mathematics, Princeton University, NJ, USA

Abstract

We prove the existence of optimal separators for intersection graphs of balls and spheres in any dimension d. One of our results is that if an intersection graph of n spheres in $\mathbb{R}^d$ has m edges, then it contains a balanced separator of size $O_d(m^{1/d}n^{1-2/d})$. This bound is best possible in terms of the parameters involved. The same result holds if the balls and spheres are replaced by fat convex bodies and their boundaries.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Graph theory

Keywords and phrases graph separators, intersection graphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.50

Related Version *Full Version*: arXiv:2603.22204

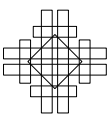
Funding This material is based in part upon work supported by the National Science Foundation under Grant No. DMS-1928930, while the authors were in residence at the Simons Laufer Mathematical Sciences Institute in Berkeley, California, during the Spring 2025 semester.
Jacob Fox: supported by NSF Awards DMS-2452737 and DMS-2154129.
Jonathan Tidor: supported in part by a Stanford Science Fellowship.

1 Introduction

For a graph $G = (V, E)$, a *balanced separator* is a vertex set $\mathcal{X} \subseteq V$ such that each connected component of $G \setminus \mathcal{X}$ contains at most $\frac{2}{3}|V|$ vertices. In 1979, Lipton and Tarjan [15] showed that n-vertex planar graphs have balanced separators of size $O(\sqrt{n})$. There are many different characterizations of planar graphs that each lead to a different extension of the Lipton–Tarjan separator theorem. For example, Wagner's theorem [23] states that a graph is planar if and only if it does not contain K_5 or $K_{3,3}$ as a minor. The family of graphs that can be embedded in a fixed surface has a finite list of forbidden minors by the Robertson–Seymour theorem [19]. Gilbert, Hutchinson, and Tarjan [9] proved that every n-vertex graph which can be embedded in a surface of genus g has a separator of size $O(\sqrt{gn})$. Later, Alon, Seymour, and Thomas [1] proved that for every graph H, every n-vertex graph that does not contain H as a minor has a separator of size at most $O_H(\sqrt{n})$.

Another important characterization of planar graphs is the Koebe–Andreev–Thurston circle packing theorem, which says that a graph is planar if and only if it is the intersection graph of interior-disjoint disks in the plane. (This means that there is a collection of interior-disjoint disks, one corresponding to each vertex, so that two vertices are connected by an edge if and only if the corresponding disks are tangent.) This leads to another way to try to extend the Lipton–Tarjan separator theorem, to intersection graphs of geometric objects. A *string graph* is an intersection graph of arcwise-connected sets in the plane, so planar graphs are string graphs. The Lipton–Tarjan separator theorem has been extended in several works studying separators for string graphs [6, 7, 12, 13, 17], and we now know that every string graph with m edges has a balanced separator of size $O(\sqrt{m})$.


© Jacob Fox and Jonathan Tidor;
licensed under Creative Commons License CC-BY 4.0



Graphs with a small balanced separator are poor expanders, and one can utilize the separator for many applications, including in fast divide-and-conquer type algorithms for many algorithmically hard graph problems (see, for example, [16]), extremal applications (see, for example, [8, 14]), and enumerative problems (see, for example, [5]).

In this paper, we are interested in finding small separators for intersection graphs of geometric objects in higher dimensions. Miller, Teng, Thurston, and Vavasis [18] proved a separator theorem for the intersection graph of balls in $\mathbb{R}^d$ in which no point is in more than k of the balls. Generalizing this, we prove that every intersection graph of n balls in $\mathbb{R}^d$ with m edges has a balanced separator of size $O_d(m^{1/d}n^{1-2/d})$. This bound is best possible in terms of the parameters involved. (Here and throughout the paper, we use the notation $O_d(\cdot)$ to hide a multiplicative constant that depends only on the dimension d.)

Proving a separator theorem for intersection graphs of spheres turns out to be more challenging as such graphs can have significantly more complicated combinatorial structure. Recently, Davies, Georgakopoulos, Hatzel, and McCarty [3] studied this problem under the constraint that the intersection graph is $K_{t,t}$-free, proving that if an intersection graph of n spheres in $\mathbb{R}^d$ is $K_{t,t}$-free, then it has a balanced separator of size $\tilde{O}_d(t^{10}n^{1-1/(2d+8)})$. We first study the problem in general. Despite their more complicated structure, we show that intersection graphs of spheres have the same size separators as intersection graphs of balls.

▶ **Theorem 1.1.** *An intersection graph of n spheres in $\mathbb{R}^d$ with m edges has a balanced separator of size $O_d(m^{1/d}n^{1-2/d})$.*

As a corollary, we show that every $K_{t,t}$-free intersection graph of n spheres has a separator of size $O_d(t^{1/d}n^{1-1/d})$ (see Corollary 3.9). This verifies a strengthening of a conjecture of Davies, Georgakopoulos, Hatzel and McCarty [3, Conjecture 20]. Both Theorem 1.1 and the corollary are best possible up to the constant factor depending on the dimension d (see Proposition 4.1). We also prove a refinement of Theorem 1.1 which gives a separator whose size depends on the degree sequence of the intersection graph. As a corollary, we prove that every $K_{t,t}$-free intersection graph of spheres in $\mathbb{R}^d$ has average degree $O_d(t)$. It follows that $K_{t,t}$-free intersection graphs of spheres in $\mathbb{R}^d$ have chromatic number $O_d(t)$.

We further study intersection graphs of other geometric objects in Section 4. We generalize our results from balls and spheres to fat convex bodies and their boundaries. Unlike the planar case, we show that both the fatness and convexity assumptions are necessary in dimensions three and larger: no non-trivial separator exists for the intersection graph of general convex bodies or general fat bodies.

Finally, as separators have many algorithmic applications, in Section 5 we show that there are efficient randomized algorithms for finding such separators of intersection graphs of balls or spheres.

2 Separators for intersection graphs of balls

Let $\mathcal{B}$ be a collection of balls in $\mathbb{R}^d$. The *intersection graph* of $\mathcal{B}$, denoted $G(\mathcal{B})$, is the graph with vertex set $\mathcal{B}$ where two balls are adjacent if they intersect. The collection $\mathcal{B}$ is *k-ply* if every point of $\mathbb{R}^d$ is contained in at most k balls. Miller, Teng, Thurston, and Vavasis [18] proved the following separator theorem for intersection graphs of balls.

▶ **Theorem 2.1** ([18])**.** *Let $\mathcal{B}$ be a k-ply collection of n balls in $\mathbb{R}^d$. Then $G(\mathcal{B})$ has a balanced separator of size $O_d(k^{1/d}n^{1-1/d})$.*

A more recent proof of this result was given by Har-Peled [10]. Using his techniques, we are able to give a more refined version of the Miller–Teng–Thurston–Vavasis separator theorem. (Similar techniques also appeared in an earlier work of Smith and Wormald [20].)

Given a collection of balls $\mathcal{B}$, for each point $x \in \mathcal{B}$, the *ply* of x, denoted $p(x)$, is the number of balls in $\mathcal{B}$ which contain x. For each ball $B \in \mathcal{B}$, the *ply* of B, denoted $p(B)$, is the maximum over $p(x)$ over all points $x \in B$. With this definition we strengthen Theorem 2.1 as follows.

▶ **Theorem 2.2.** *Let $\mathcal{B}$ be a collection of n balls in $\mathbb{R}^d$. Then $G(\mathcal{B})$ has a balanced separator of size*

$$O_d\left(\left(\sum_{B\in\mathcal{B}} p(B)^{\frac{1}{d-1}}\right)^{1-\frac{1}{d}}\right).$$

Since every ball has ply at most one more than its degree in the intersection graph, we have the following corollary which bounds the size of the separator in terms of the degree sequence of the graph, or simply its number of edges.

▶ **Corollary 2.3.** *Let $\mathcal{B}$ be a collection of n balls in $\mathbb{R}^d$. Then $G(\mathcal{B})$ has a balanced separator of size*

$$O_d\left(\left(\sum_{B\in\mathcal{B}} \deg(B)^{\frac{1}{d-1}}\right)^{1-\frac{1}{d}}\right).$$

In particular, if $G(\mathcal{B})$ has m edges, then it has a balanced separator of size $O_d(m^{1/d}n^{1-2/d})$.

The second claim follows from the first via an application of Hölder's inequality since

$$\sum_{B\in\mathcal{B}} \deg(B)^{\frac{1}{d-1}} \le \left(\sum_{B\in\mathcal{B}} \deg(B)\right)^{\frac{1}{d-1}} \left(\sum_{B\in\mathcal{B}} 1\right)^{\frac{d-2}{d-1}} = (2m)^{\frac{1}{d-1}} n^{\frac{d-2}{d-1}}.$$

To construct a balanced separator, we first construct a separator with slightly worse quantitative properties and then iterate a bounded number of times to find the desired separator.

▶ **Definition 2.4.** *For a graph $G = (V, E)$ and $c < 1$, a* c-balanced separator *is a vertex set $\mathcal{X} \subseteq V$ such that each connected component of $G \setminus \mathcal{X}$ contains at most $c|V|$ vertices.*

▶ **Lemma 2.5.** *Suppose graph $G = (V, E)$ is such that every induced subgraph has a c-balanced separator of size at most s. Then G has a balanced separator of size at most $\lceil \log_c(2/3) \rceil s$.*

Proof. Set $V_1 = V$. Given a vertex set V_i, let $\mathcal{X}_i$ be a c-balanced separator of $G[V_i]$ of size at most s. Define V_{i+1} to be the set of vertices comprising the largest connected component of $G[V_i] \setminus \mathcal{X}_i$. By definition, $|V_{i+1}| \le c|V_i|$. Furthermore, each other connected components of $G[V_i] \setminus \mathcal{X}_i$ contains at most $\frac{1}{2}|V_i| \le \frac{2}{3}|V|$ vertices. Halt the process at the first $i = r$ satisfying $|V_{r+1}| \le \frac{2}{3}|V|$, so $r \le \lceil \log_c(2/3) \rceil$. Defining $\mathcal{X} = \mathcal{X}_1 \cup \cdots \cup \mathcal{X}_r$, we have $|\mathcal{X}| \le rs \le \lceil \log_c(2/3) \rceil s$. Furthermore, every connected component of $G \setminus \mathcal{X}$ is either equal to V_{r+1}, or is a connected component of $G[V_i] \setminus \mathcal{X}_i$ other than V_{i+1} for some i. Since all of these sets have size at most $\frac{2}{3}|V|$, this implies that $\mathcal{X}$ is a balanced separator for G. ◀

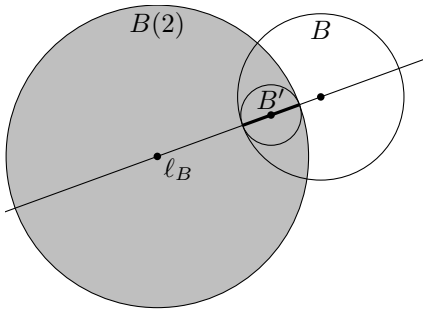


Figure 1 Constructing $\mathcal{B}'$ from $\mathcal{B}$.

Proof of Theorem 2.2. We will give a randomized procedure that produces a $(1-5^{-d})$-balanced separator $\mathcal{X}$ for $G(\mathcal{B})$ of the desired size. By Lemma 2.5, losing another constant factor depending only on d, this suffices to prove the desired result.

Let B_0 be the ball in $\mathbb{R}^d$ with smallest radius that contains at least $5^{-d}n$ centers of balls of $\mathcal{B}$. (B_0 is not necessarily an element of $\mathcal{B}$.) By perturbing B_0 slightly, one can see that B_0 contains exactly $\lceil 5^{-d}n \rceil$ centers. After translating and rescaling the problem appropriately, assume that $B_0 = B(1)$, the origin-centered ball of radius 1.

For some $r \in [1,2]$, let $\mathcal{X}$ be the set of balls $B \in \mathcal{B}$ which intersect the origin-centered sphere of radius r. We claim that for any $r \in [1,2]$, the set $\mathcal{X}$ is a $(1-5^{-d})$-balanced separator for $G(\mathcal{B})$. To see this, note that each connected component of $\mathcal{B} \setminus \mathcal{X}$ consists of balls which are either entirely inside $B(r)$ or entirely outside $B(r)$. Since $B(1) \subseteq B(r)$ contains $\lceil 5^{-d}n \rceil$ centers, there are at most $(1-5^{-d})n$ balls entirely outside $B(r)$. Now $B(2) \supseteq B(r)$ can be covered by 4^d translates of $B(1)$ which, by the minimality of B_0, each contain at most $\lceil 5^{-d}n \rceil$ centers. Thus $B(2)$ contains at most $4^d \lceil 5^{-d}n \rceil \leq (1-5^{-d})n$ centers, so there are at most this many balls entirely inside $B(r)$.

We now show that there is a choice of $r \in [1,2]$ so that $\mathcal{X}$ is small. To do this, pick $r \in [1,2]$ uniformly at random. We now bound the expected size of $\mathcal{X}$. We do this by modifying $\mathcal{B}$ by shrinking those balls that intersect $B(2)$ so that they lie entirely in $B(2)$. For each $B \in \mathcal{B}$ which intersects $B(2)$, let ℓ_B be the line that passes through the origin and the center of B. Let B' be the ball contained in $B(2)$ whose center lies on ℓ_B and which satisfies $\ell_B \cap B' = \ell_B \cap B \cap B(2)$. (See Figure 1.) Define $\mathcal{B}'$ to be the set of these balls B' (the balls in $\mathcal{B}$ that do not intersect $B(2)$ do not have any corresponding ball in $\mathcal{B}'$).

Since each ball $B' \in \mathcal{B}'$ lies inside the corresponding ball $B \in \mathcal{B}$, clearly we have $p_{\mathcal{B}'}(B') \leq p_{\mathcal{B}}(B)$. Write $p(x)$ for the number of balls of $\mathcal{B}'$ which contain the point x. Then $p_{\mathcal{B}'}(B) = \max_{x \in B} p(x)$. We now have

$$\begin{aligned}\operatorname{vol}(B(2)) &\geq \operatorname{vol}\left(\bigcup_{B \in \mathcal{B}'} B\right) = \int_{\bigcup_{B \in \mathcal{B}'} B} p(x) \frac{\mathrm{d}x}{p(x)} = \sum_{B \in \mathcal{B}'} \int_B \frac{\mathrm{d}x}{p(x)} \\ &\geq \sum_{B \in \mathcal{B}'} \int_B \frac{\mathrm{d}x}{p_{\mathcal{B}'}(B)} = \sum_{B \in \mathcal{B}'} \frac{\operatorname{vol}(B)}{p_{\mathcal{B}'}(B)}.\end{aligned}$$

Thus, dividing by $\operatorname{vol}(B(1))$, we conclude that

$$2^d \geq \sum_{B \in \mathcal{B}'} \frac{r(B)^d}{p_{\mathcal{B}'}(B)}.$$

Now for each $B \in \mathcal{B}$, the probability that B is in the separator is at most the length of the interval $\ell_B \cap B \cap B(2)$. In particular, the probability is 0 if B does not intersect $B(2)$. Now if B does intersect $B(2)$, it corresponds to a ball $B' \in \mathcal{B}'$ so that $\ell_B \cap B \cap B(2) = \ell_B \cap B'$ is an interval of length $2r(B')$. Thus by Hölder's inequality we conclude

$$\begin{aligned}\mathbb{E}[|\mathcal{X}|] &\leq 2\sum_{B\in\mathcal{B}'} r(B) \leq 2\left(\sum_{B\in\mathcal{B}'} \frac{r(B)^d}{p_{\mathcal{B}'}(B)}\right)^{1/d} \cdot \left(\sum_{B\in\mathcal{B}'} p_{\mathcal{B}'}(B)^{1/(d-1)}\right)^{(d-1)/d} \\ &\leq 4\left(\sum_{B\in\mathcal{B}'} p_{\mathcal{B}'}(B)^{1/(d-1)}\right)^{(d-1)/d} \leq 4\left(\sum_{B\in\mathcal{B}} p_{\mathcal{B}}(B)^{1/(d-1)}\right)^{(d-1)/d}. \qquad \blacktriangleleft\end{aligned}$$

▶ Remark 2.6. Clearly Theorem 2.2 implies Theorem 2.1. We point out that even the weakest version of Corollary 2.3 is sufficient to imply Theorem 2.1. To see this, let $\mathcal{B}$ be a k-ply collection of balls in $\mathbb{R}^d$. Then the intersection graph $G(\mathcal{B})$ is $O_d(k)$-degenerate. (Recall that a graph is D-degenerate if there exist an ordering of the vertices so that each vertex is adjacent to at most D vertices which appear earlier in the ordering.) To see this, order the balls by radius. One can check that any ball B_j which intersects B_i and has radius satisfying $r_j \geq r_i$ must occupy at least a $(1/3)^d$-fraction of B_i', defined to be the dilate of B_i by a factor of 3. The k-ply assumption shows that there are at most $3^d k$ such balls B_j. Thus $G(\mathcal{B})$ has at most $3^d kn$ edges so Theorem 2.1 follows from Corollary 2.3.

3 Separators for intersection graphs of spheres

The main difficulty in generalizing the results of the previous section to intersection graphs of spheres is that the notion of ply is not useful for sphere intersection graphs. In particular, one can nest spheres so that the corresponding collection of balls would have large ply yet the sphere intersection graph has few or no edges. Furthermore, there are dense graphs (in particular, complete bipartite graphs) that are intersection graphs of spheres but are 2-ply in the sense that no point lies on more than two spheres. (To see this, take two nested families of spheres that are arranged so that every sphere from one family intersects every sphere from the other family.) This shows that no analogue of the argument in Remark 2.6 applies to sphere intersection graphs. Despite these difficulties, we show that separators of the same size exist.

▶ **Theorem 3.1.** *Let $\mathcal{S}$ be a collection of n spheres in $\mathbb{R}^d$. Then $G(\mathcal{S})$ has a balanced separator of size at most*

$$O_d\left(\left(\sum_{S\in\mathcal{S}} \deg(S)^{\frac{1}{d-1}}\right)^{1-\frac{1}{d}}\right).$$

As with Corollary 2.3, this result implies Theorem 1.1 by an application of Hölder's inequality. We start with an elementary lemma that is needed for the proof.

▶ **Lemma 3.2.** *Let $\mathcal{S}$ be a collection of spheres in $\mathbb{R}^d$. Suppose that $G(\mathcal{S})$ has maximum degree at most Δ and that there exists a point $p \in \mathbb{R}^d$ contained in the interior of at least $2s$ spheres. Then there exists a set $\mathcal{X} \subseteq \mathcal{S}$ with $|\mathcal{X}| \leq \Delta$ such that every connected component of $G(\mathcal{S}) \setminus \mathcal{X}$ has size at most $|\mathcal{S}| - s$.*

Proof. Let $\mathcal{S}' \subseteq \mathcal{S}$ be the set of spheres containing p in their interior. For any two spheres $S, S' \in \mathcal{S}'$, either they intersect or one contains the other in its interior. Consider the partial order on $\mathcal{S}'$ defined by containment. Take any linear extension of this poset and let $S_0 \in \mathcal{S}'$

be the median element of this linear extension. This means that there are at least $s-1$ spheres which either intersect S_0 or are contained in its interior and there are at least $s-1$ spheres which either intersect S_0 or contain it in their interior.

Now let $\mathcal{X} = N(S_0)$, the neighborhood of S_0 in $G(\mathcal{S})$, i.e., the set of spheres which intersect S_0. We claim that every connected component of $G(\mathcal{S}) \setminus \mathcal{X}$ has size at most $|\mathcal{S}| - s$. To see this, note that no sphere in $\mathcal{S} \setminus \mathcal{X}$ intersects S_0, so any connected component of $G(\mathcal{S}) \setminus \mathcal{X}$ consists of spheres either entirely inside S_0 or entirely outside S_0. However, we showed that there are at least s spheres that are not of the first type (S_0 and the $s-1$ spheres which follow it in the linear extension) and at least s spheres that are not of the second type. Thus each connected component of $G(\mathcal{S}) \setminus \mathcal{X}$ has size at most $|\mathcal{S}| - s$. ◀

Proof of Theorem 3.1. Set

$$\Sigma = C_d \left(\sum_{S \in \mathcal{S}} \deg(S)^{\frac{1}{d-1}} \right)^{1-\frac{1}{d}}.$$

Choosing $c_d > 0$ small in terms of d and C_d large in terms of c_d, we will construct a $(1-c_d)$-balanced separator for $G(\mathcal{S})$ of size at most Σ. By Lemma 2.5, losing another constant factor depending only on d, this suffices to prove the desired result. Assume $\Sigma < n$, since the result is trivial otherwise.

Through the proof, if we have not already found the desired separator, we will define sets $\mathcal{X}_i \subset \mathcal{S}$ for $i = 1, 2, 3, 4$ with $|\mathcal{X}_i| \leq \Sigma/4$ for each i so that $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3 \cup \mathcal{X}_4$ is the desired separator.

Step 1: First, define $\mathcal{X}_1 \subset \mathcal{S}$ to be the $\Sigma/4$ vertices of highest degree in $G(\mathcal{S})$. Let Δ be the maximum degree of the resulting graph $G(\mathcal{S}) \setminus \mathcal{X}_1$. We claim that $\Delta \leq \Sigma/4$ and $\Delta \leq c_d n$. If the first does not hold, then $G(\mathcal{S})$ would have at least $\Sigma/4$ vertices of degree at least $\Sigma/4$, meaning that

$$\Sigma = C_d \left(\sum_{S \in \mathcal{S}} \deg(S)^{1/(d-1)} \right)^{1-1/d} \geq C_d \left((\Sigma/4)(\Sigma/4)^{1/(d-1)} \right)^{1-1/d} = \frac{C_d}{4}\Sigma.$$

Clearly this is impossible for $C_d > 4$. Similarly, if the second fails then $G(\mathcal{X})$ would have at least $\Sigma/4$ vertices of degree at least $c_d n$, meaning that

$$\Sigma \geq C_d \left((\Sigma/4)(c_d n)^{1/(d-1)} \right)^{1-1/d} = \frac{C_d c_d^{1/d}}{4^{1-1/d}} \Sigma^{1-1/d} n^{1/d} > \frac{C_d c_d^{1/d}}{4^{1-1/d}} \Sigma,$$

where the final inequality holds since $\Sigma < n$. Again, this is impossible for C_d sufficiently large in terms of c_d.

Step 2: Write $\mathcal{S}_1 = \mathcal{S} \setminus \mathcal{X}_1$. The goal of this step is to pass to a subset $\mathcal{S}_2 \subseteq \mathcal{S}_1$ such that every sphere $S \in \mathcal{S}_2$ contains fewer than $4c_d n$ centers of spheres in $\mathcal{S}_2$. Let $S_0 \in \mathcal{S}_1$ be a sphere of minimal radius that contains $4c_d n$ centers of spheres in $\mathcal{S}_1$. (If no such sphere exists, taking $\mathcal{S}_2 = \mathcal{S}_1$ completes this step.) Define $\mathcal{S}_{\supseteq S_0} \subseteq \mathcal{S}_1$ to be the set of spheres which contain S_0 entirely within their interior. First suppose that $|\mathcal{S}_{\supseteq S_0}| \geq 2c_d n$. In this case we can apply Lemma 3.2 to find a separator $\mathcal{X}_2 \subset \mathcal{S}_1$ such that $|\mathcal{X}_2| \leq \Delta \leq \Sigma/4$ and each connected component of $G(\mathcal{S}_1) \setminus \mathcal{X}_2$ has size at most $(1-c_d)n$. Then $\mathcal{X}_1 \cup \mathcal{X}_2$ is a $(1-c_d)$-balanced separator for $G(\mathcal{S})$ of size at most $\Sigma/2$.

Thus we can assume that $|\mathcal{S}_{\supseteq S_0}| < 2c_d n$. In this case, define $\mathcal{X}_2 = N(S_0)$, the neighborhood of S_0 in $G(\mathcal{S})$. Define $\mathcal{S}_2 \subseteq \mathcal{S}_1$ to be the set of spheres which are entirely contained within S_0. Clearly $|\mathcal{X}_2| \leq \Delta \leq \Sigma/4$.

▷ Claim 3.3. If $G(\mathcal{S}) \setminus (\mathcal{X}_1 \cup \mathcal{X}_2)$ has a connected component of size more than $(1 - c_d)n$, then it is also a connected component of $G(\mathcal{S}_2)$.

Proof. Every connected component of $G(\mathcal{S}) \setminus (\mathcal{X}_1 \cup \mathcal{X}_2)$ is either made up of spheres entirely inside S_0 or entirely outside S_0. The former are also connected components of $G(\mathcal{S}_2)$ while the latter are not. There are at least $4c_d n$ centers of spheres of $\mathcal{S}_1$ inside of S_0. Since at most $2c_d n$ of these correspond to spheres surrounding S_0 and at most Δ correspond to spheres which intersect S_0, we see that there are at least $2c_d n - \Delta \geq c_d n$ spheres inside S_0. Therefore any connected component outside of S_0 has size at most $(1 - c_d)n$. ◁

We now construct a separator for $G(\mathcal{S}_2)$. We know that every sphere in $\mathcal{S}_2$ contains fewer than $4c_d n$ centers of spheres in $\mathcal{S}_2$. We can also assume that $|\mathcal{S}_2| \geq (1 - c_d)n$, otherwise $\mathcal{X}_1 \cup \mathcal{X}_2$ is already the desired $(1 - c_d)$-balanced separator.

Step 3: Let B_{in} be a ball of minimal radius which contains at least $4c_d n$ centers of spheres in $\mathcal{S}_2$. Since we showed that no sphere in $\mathcal{S}_2$ contains $4c_d n$ centers, we see that no sphere in $\mathcal{S}_2$ contains B_{in} entirely in its interior. This means that each of the $4c_d n$ spheres whose centers lie in B_{in} also intersect B_{in}.

Write R for the radius of B_{in} and define B_{out} to be the ball with the same center and radius $2R$. By the minimality of B_{in}, we see that B_{out} contains at most $4^d \lceil 4c_d n \rceil$ centers. In particular, choosing c_d sufficiently small, there are at least $c_d n$ sphere centers outside of B_{out}. Let S_{rand} be a sphere with the same center as B_{in} and B_{out} whose radius is chosen uniformly at random in $[R, 2R]$.

For a sphere $S \in \mathcal{S}$, write $r(S)$ for the radius of S. Write $\tilde{r}(S)$ for the radius of the spherical cap formed by intersecting the ball bounded by S with S_{rand}. (The radius of a spherical cap is the minimal $\tilde{r}(S)$ so that the cap is contained in a ball of radius $\tilde{r}(S)$.)

Choose some C'_d sufficiently large in terms of d and let $\mathcal{X}_3$ consist of the spheres $S \in \mathcal{S}_2$ which intersect S_{rand} and which satisfy $\tilde{r}(S) \leq C'_d R(\deg(S)/\Sigma)^{1/(d-1)}$.

▷ Claim 3.4. $\mathbb{E}[|\mathcal{X}_3|] \leq \Sigma/4$.

Proof. We need to upper bound the probability that a sphere S intersects S_{rand} in a spherical cap of radius at most $\tilde{r}$. If $\tilde{r} \leq r(S)$, the probability is upper bounded by

$$\frac{2\left(r(S) - \sqrt{r(S)^2 - \tilde{r}^2}\right)}{R} \leq \frac{2\tilde{r}}{R}.$$

To see this, note that this event is the same as the event that S_{rand} intersects the line connecting the center of B_{in} and the center of S in one of the bold segments in Figure 2.

Now if $\tilde{r} > r(S)$, the probability we need to upper bound is just the probability that S_{rand} and S intersect. This quantity is at most $2r(S)/R < 2\tilde{r}/R$.

Since $\mathcal{X}_3$ is defined with the cutoff $\tilde{r} = C'_d R(\deg(S)/\Sigma)^{1/(d-1)}$, for each $S \in \mathcal{S}_2$ we have the bound

$$\Pr\left[S \text{ intersects } S_{\text{rand}} \text{ and } \tilde{r}(S) \leq C'_d R(\deg(S)/\Sigma)^{1/(d-1)}\right] \leq 2C'_d(\deg(S)/\Sigma)^{1/(d-1)}.$$

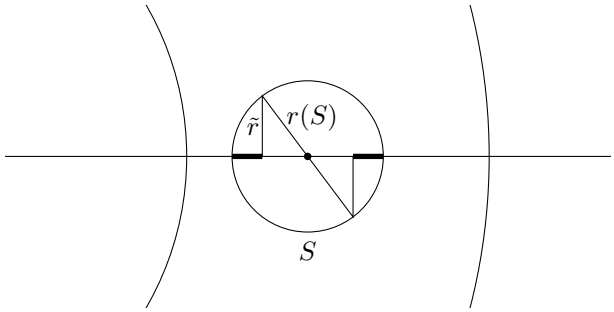


Figure 2 Bounding $\Pr[S \in \mathcal{X}_3]$.

This implies that

$$\mathbb{E}[|\mathcal{X}_3|] \leq \sum_{S \in \mathcal{S}_2} 2C'_d(\deg(S)/\Sigma)^{1/(d-1)} = \frac{2C'_d}{\Sigma^{1/(d-1)} C_d^{d/(d-1)}} \Sigma^{d/(d-1)} \leq \Sigma/4,$$

where the equality is by the definition of Σ and the last inequality holds for C_d chosen appropriately large in terms of C'_d. ◁

Fix some choice of S_{rand} for which $|\mathcal{X}_3| \leq \Sigma/4$.

Step 4: Define $\mathcal{S}_3 = \mathcal{S}_2 \setminus \mathcal{X}_3$. We perform the following iterative process. Start with $\mathcal{X}_4 = \emptyset$ and $\mathcal{S}_4 = \mathcal{S}_3$. We will define spheres $S_1, S_2, \ldots, S_\ell$. Suppose we have defined $S_1, \ldots, S_{i-1}$. Define S_i to be a sphere in $\mathcal{S}_4$ which intersects S_{rand} and has maximal radius among such spheres. Place the neighborhood $N(S_i)$ in $\mathcal{X}_4$ and remove all spheres which intersect the ball bounded by S_i from $\mathcal{S}_4$ (in particular we remove S_i and $N(S_i)$ from $\mathcal{S}_4$). Repeat until no more spheres in $\mathcal{S}_4$ intersect S_{rand}.

We claim that the balls bounded by $S_1, S_2, \ldots, S_\ell$ are disjoint. For $i < j$ we know that $r(S_i) \geq r(S_j)$ and the ball bounded by S_i is disjoint from the sphere S_j. The latter implies that S_i, S_j do not intersect and that S_i does not contain S_j; the former implies that S_j does not contain S_i.

▷ Claim 3.5. $|\mathcal{X}_4| \leq \Sigma/4$.

Proof. By the definition of $\mathcal{X}_3$, every sphere $S \in \mathcal{S}_3$ which intersects S_{rand} also satisfies $\tilde{r}(S) > C'_d R(\deg(S)/\Sigma)^{1/(d-1)}$. Thus

$$|\mathcal{X}_4| \leq \sum_{i=1}^{\ell} \deg(S_i) < \sum_{i=1}^{\ell} \Sigma \left(\frac{\tilde{r}(S_i)}{C'_d R} \right)^{d-1}.$$

Now since the balls bounded by $S_1, \ldots, S_\ell$ are disjoint, we have

$$\sum_{i=1}^{\ell} \tau_{d-1} \tilde{r}(S_i)^{d-1} \leq \sigma_d (2R)^{d-1}$$

where τ_k, σ_k denote the k-volume of the unit ball in $\mathbb{R}^k$ and the $(k-1)$-volume of the unit sphere in $\mathbb{R}^k$, respectively. (In the above inequality, we use the fact that the $(d-1)$-volume of a spherical cap of radius $\tilde{r}$ on any sphere in $\mathbb{R}^d$ is at least the $(d-1)$-volume of a ball of radius $\tilde{r}$ in $\mathbb{R}^{d-1}$.) Thus we conclude that

$$|\mathcal{X}_4| \leq \Sigma \frac{2^{d-1}\sigma_d}{C_d'^{d-1}\tau_{d-1}} \leq \Sigma/4,$$

where the last inequality holds for C'_d chosen sufficiently large in terms of d. ◁

▷ Claim 3.6. $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3 \cup \mathcal{X}_4$ is a $(1 - c_d)$-balanced separator for $G(\mathcal{S})$.

Proof. By Claim 3.3, it suffices to show that each connected component of $G(\mathcal{S}_2) \setminus (\mathcal{X}_3 \cup \mathcal{X}_4)$ has size at most $(1 - c_d)n$. Note that any sphere in $\mathcal{S}_2 \setminus (\mathcal{X}_3 \cup \mathcal{X}_4)$ which crosses S_{rand} must lie in the ball bounded by one of the S_i. Furthermore, no sphere in $\mathcal{S}_2 \setminus (\mathcal{X}_3 \cup \mathcal{X}_4)$ crosses any of the S_i. Therefore any connected component of $G(\mathcal{S}_2) \setminus (\mathcal{X}_3 \cup \mathcal{X}_4)$ must either lie entirely within some sphere S_i, entirely within S_{rand}, or entirely outside of S_{rand}. We chose $\mathcal{S}_2$ so that any sphere in $\mathcal{S}_2$ contains at most $4c_d n$ centers of spheres in $\mathcal{S}_2$. Thus the first type of connected component has size at most $4c_d n$. For the second type, note that S_{rand} is contained within B_{out} which we argued earlier contains at most $4^d \lceil 4c_d n \rceil < (1 - c_d)n$ centers of spheres in $\mathcal{S}_2$. For the third type of connected component, we know that there are at least $4c_d n$ spheres which intersect B_{in}. Since B_{in} is contained in the interior of S_{rand}, we see that there are at most $(1 - 4c_d)n$ spheres outside of S_{rand}, giving the desired bound on the third type of connected component. ◁

Since $|\mathcal{X}| \leq \Sigma$, this completes the proof. ◀

Intersection graphs of spheres are an example of semialgebraic graphs. Using known bounds on the number of edges in $K_{t,t}$-free semialgebraic graphs [21], we deduce the following weak bound for separators in $K_{t,t}$-free sphere intersection graphs.

▶ **Corollary 3.7.** *Let $\mathcal{S}$ be a collection of n spheres in $\mathbb{R}^d$. If $G(\mathcal{S})$ is $K_{t,t}$-free, then it has a balanced separator of size $O_d(t^{2/d(d+2)} n^{1-2/d(d+2)})$.*

Proof. Represent the sphere centered at $x \in \mathbb{R}^d$ with radius r by the point $(x, r) \in \mathbb{R}^{d+1}$. Now two spheres (x, r) and (x', r') intersect if and only if $(r - r')^2 \leq \|x - x'\|^2 \leq (r + r')^2$. These are polynomial inequalities of degree $O(1)$, meaning that $G(\mathcal{S})$ can be represented by a semialgebraic graph in $\mathbb{R}^{d+1}$ of total degree $O(1)$. (See [21, Definition 1.1] for the full definition.)

By [21, Theorem 1.6] (a bound on the Zarankiewicz problem for semialgebraic graphs), if $G(\mathcal{S})$ is $K_{t,t}$-free, then it has $O_d(t^{2/(d+2)} n^{2(d+1)/(d+2)})$ edges. Applying Theorem 1.1 with this bound shows that $G(\mathcal{S})$ has a balanced separator of size

$$O_d\left(\left(t^{2/(d+2)} n^{2(d+1)/(d+2)}\right)^{1/d} n^{1-2/d}\right) = O_d(t^{2/d(d+2)} n^{1-2/d(d+2)}).$$ ◀

We can bootstrap this weak bound to get a better bound on the number of edges in a $K_{t,t}$-free sphere intersection graph. This follows from [6, Lemma 8] which turns a bound on the size of separators into a bound on the number of edges for any hereditary family of graphs. (See [8, Section 3] for another application of this method.)

▶ **Corollary 3.8.** *Let $\mathcal{S}$ be a collection of n spheres in $\mathbb{R}^d$. If $G(\mathcal{S})$ is $K_{t,t}$-free, then it has $O_d(tn)$ edges.*

Proof. We aim to apply [6, Lemma 8]. This result requires a monotone function ϕ so that any n-vertex $K_{t,t}$-free intersection graph of spheres in $\mathbb{R}^d$ contains a balanced separator of size at most $n\phi(n)$. By Corollary 3.7, we can take $\phi(n) = C_d(t/n)^{2/d(d+2)}$ where C_d is the constant in that result. Clearly $\phi(n)$ is monotone.

Set $n_0 = (12C_d)^{d(d+2)/2} t$ so that $\phi(n_0) = 1/12$. Then [6, Lemma 8] implies that every $K_{t,t}$-free intersection graph of spheres in $\mathbb{R}^d$ on $n \geq n_0$ vertices has at most $Cn_0 n/2$ edges where

$$C = \prod_{i=0}^{\infty} \left(1 + \phi\left(\lceil (4/3)^i n_0 \rceil\right)\right).$$

We can bound

$$C \leq \exp\left(\sum_{i=0}^{\infty} \phi\left(\lceil (4/3)^i n_0 \rceil\right)\right) \leq \exp\left(2C_d \sum_{i=0}^{\infty} \left(\left(\frac{3}{4}\right)^i \frac{t}{n_0}\right)^{2/d(d+2)}\right).$$

As $n_0 = O_d(t)$, this geometric series sums to $O_d(1)$, showing that $C = O_d(1)$. This gives the desired bound on the number of edges when $n \geq n_0$.

Note that if $n < n_0$, then we can trivially bound the number of edges by $n^2 < n_0 n = O_d(tn)$. ◀

Combining Theorem 1.1 and Corollary 3.8, we immediately conclude the following strengthening of Corollary 3.7. This result improves upon the sphere separator result of Davies–Georgakopoulos–Hatzel–McCarty [3, Theorem 1], proving a strengthening of [3, Conjecture 20].

▶ **Corollary 3.9.** *Let $\mathcal{S}$ be a collection of n spheres in $\mathbb{R}^d$. If $G(\mathcal{S})$ is $K_{t,t}$-free, then it has a balanced separator of size $O_d(t^{1/d}n^{1-1/d})$.*

▶ Remark 3.10. The same proof also gives Corollaries 3.8 and 3.9 for intersection graphs of balls. However these results actually have significantly easier proofs. For example, let $\mathcal{B}$ be a collection of balls in $\mathbb{R}^d$. If $\mathcal{B}$ is $(2t-1)$-ply, then Theorem 2.1 already implies that $G(\mathcal{B})$ has a balanced separator of size $O_d(t^{1/d}n^{1-1/d})$. Otherwise, there is a point in $2t$ balls of $\mathcal{B}$, implying that $G(\mathcal{B})$ contains K_{2t} which contains $K_{t,t}$.

4 Further results

We start by giving a construction that shows that our main results are asymptotically optimal. We then generalize these results to a larger class of geometric objects.

▶ **Proposition 4.1.** *For each $n, d \geq 1$ and $n \leq m \leq \binom{n}{2}$, there exists an intersection graph of n balls or spheres in $\mathbb{R}^d$ with at most m edges for which every balanced separator has size at least $\Omega_d(m^{1/d}n^{1-2/d})$.*

Proof. Let $V = \{1, \ldots, k\}^d$ for $k = \lfloor n^{1/d} \rfloor$. Define $G = (V, E)$ where $x, y \in V$ are adjacent if $\|x - y\| \leq r$. Choosing $1 \leq r = \Theta_d((m/n)^{1/d})$ appropriately, we can ensure that G has $\Theta_d(m)$ edges. Note that G is both the intersection graph of the radius $r/2$ balls centered at the points of V and the intersection graph of the radius $r/2$ spheres centered at the points of V. We will show that every balanced separator of G has size at least $\Omega_d(m^{1/d}n^{1-2/d})$.

We first handle the case when $r = 1$. Suppose that $\mathcal{X}$ is a balanced separator with $|\mathcal{X}| \leq k^d/3$. Let $A \subset V$ be the union of some connected components of $G \setminus \mathcal{X}$, satisfying $k^d/3 \leq |A| \leq 2k^d/3$. When $r = 1$, the graph G is known as the grid graph on $\{1, \ldots, k\}^d$. Then the isoperimetric inequality on the grid graph [2] implies that there are at least k^{d-1} edges between A and $V \setminus A$. However, all such edges must have one endpoint in $\mathcal{X}$. Since the maximum degree of G is $2d$, we conclude that $|\mathcal{X}| \geq k^{d-1}/2d \geq \Omega_d(n^{1-1/d})$, as desired.

To handle the general case, let $r' = r/(2\sqrt{d})$. For simplicity, assume that r' is an integer that divides k, say $k = r'k'$. Divide $\{1, \ldots, k\}^d$ into $(k')^d$ subcubes of side length r'. Note that every vertex in one subcube is adjacent in G to every vertex in an adjacent subcube. Similar to before, suppose that $\mathcal{X}$ is a balanced separator with $|\mathcal{X}| \leq k^d/12$ and A is a union of connected components of $G \setminus \mathcal{X}$ with $k^d/3 \leq |A| \leq 2k^d/3$. Write $B = V \setminus (\mathcal{X} \cup A)$. We know $|B| \geq k^d/4$. Define $A' \subseteq \{1, \ldots, k'\}^d$ where a point lies in A' if the corresponding

$r' \times \cdots \times r'$ subcube of $\{1, \dots, k\}^d$ contains at least one point of A. Define $B' \subseteq \{1, \dots, k'\}^d$ similarly. First note that A' and B' are disjoint – if they shared a vertex then A, B would contain points in the same subcube which have distance at most $\sqrt{d}r' < r$.

Write $X' = \{1, \dots, k'\}^d \setminus (A' \cup B')$. Note that $|A'| \geq |A|/(r')^d \geq (k')^d/3$ and similarly $|B'| \geq |B|/(r')^d \geq (k')^d/4$. Thus $|A'| \leq (k')^d - |B'| \leq 3(k')^d/4$. Applying the isoperimetric inequality to the grid graph on $\{1, \dots, k'\}^d$, we conclude that there are at least $(k')^{d-1}$ edges between A' and $B' \cup X'$. However, there are no edges between A', B' because any two points in two adjacent subcube are at distance at most $2\sqrt{d}r' = r$. Thus we conclude that $|X'| \geq (k')^{d-1}/(2d)$. Note though that a point lies in X' only if the corresponding subcube lies entirely in $\mathcal{X}$. We thus see that $|\mathcal{X}| \geq (r')^d(k')^{d-1}/(2d) = \Theta_d(m^{1/d}n^{1-2/d})$. ◀

Our main results on ball and sphere separators, Theorems 2.2 and 3.1, hold for more general classes of geometric objects. We say that a set $B \subset \mathbb{R}^d$ is *C-fat* if B contains a ball of radius r and is contained inside a ball of radius Cr for some r. Our results hold for fat convex bodies and for the boundaries of fat convex bodies. (See [11, 4] for related results under slightly different hypotheses.)

▶ **Theorem 4.2.** *Let $\mathcal{B}$ be a collection of n C-fat convex bodies in $\mathbb{R}^d$. Then $G(\mathcal{B})$ has a balanced separator of size at most*

$$O_{d,C}\left(\left(\sum_{B \in \mathcal{B}} p(B)^{\frac{1}{d-1}}\right)^{1-\frac{1}{d}}\right).$$

In particular, if $G(\mathcal{B})$ has m edges, then it has a balanced separator of size $O_d(m^{1/d}n^{1-2/d})$.

Proof. For each $B \in \mathcal{B}$, pick an arbitrary point $p \in \mathrm{int}(B)$ that we call the *center* of B. With this definition, we proceed with the proof of Theorem 2.2 in the same way. The same randomized procedure produces a $(1 - 5^{-d})$-balanced separator.

For each $B \in \mathcal{B}$, let $x_B \subset \mathbb{R}_{\geqslant 0}$ be the set of radii r so that the origin-centered sphere of radius r intersects B. We wish to show that if $x_B \cap [1, 2]$ is non-empty, B contains a genuine ball B' which is contained in $B(2)$ such that $r(B') \geq \Omega_C(|x_B \cap [1, 2]|)$.

First, if $x_B \subseteq [0, 2]$, then B is contained in $B(2)$. Now B contains a point of norm $\max(x_B)$ and a point of norm $\min(x_B)$, i.e., it contains two points at distance at least $\max(x_B) - \min(x_B) \geq |x_B|$. This means that any ball containing B must have radius at least $|x_B|/2$. By hypothesis, B contains a ball of radius $|x_B|/2C$. Letting B' be this ball, we are done since in this case $B' \subseteq B$ is contained in $B(2)$.

Now suppose $x_B \not\subseteq [0, 2]$, but $x_B \cap [0, 2] \neq \emptyset$. By the same argument, B contains some ball (say $\tilde{B}$) of radius $(\max(x_B) - \min(x_B))/2C$ centered at some point p with $\|p\| \leq \max(x_B)$. By assumption, B contains a point q with $\|q\| = \min(x_B) \leq 2$. (See Figure 3.) Then the convexity of B implies that B contains the convex hull of $\{q\} \cup \tilde{B}$. Elementary geometry implies that this convex hull contains a ball B' of radius $\frac{2 - \min(x_B)}{2C+1} \geq \frac{|x_B \cap [1,2]|}{2C+1}$ that is contained in $B(2)$. (To see this, dilate the convex hull of $\{q\} \cup \tilde{B}$ by a factor of $(2 - \|q\|)/(\|p\| + r(\tilde{B}) - \|q\|)$ so that it lies within $B(2)$.)

Thus we have constructed a set $\mathcal{B}'$ contained in $B(2)$ so that $\mathbb{E}[|\mathcal{X}|] \geq \Omega_C\left(\sum_{B \in \mathcal{B}'} r(B)\right)$. Note that $\mathcal{B}'$ consists of genuine balls, so the rest of the proof applies unchanged. ◀

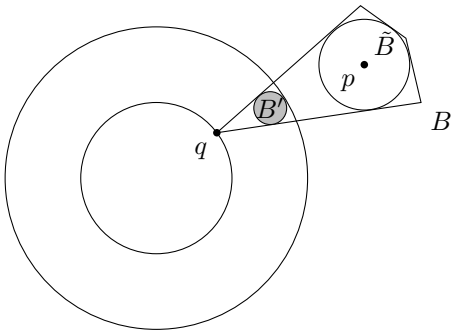


Figure 3 The construction of B' inside of $B \cap B(2)$.

▶ **Theorem 4.3.** *Let $\mathcal{S}$ be a collection of n boundaries of C-fat convex bodies in $\mathbb{R}^d$. Then $G(\mathcal{S})$ has a balanced separator of size at most*

$$O_{d,C}\left(\left(\sum_{S\in\mathcal{S}} \deg(S)^{\frac{1}{d-1}}\right)^{1-\frac{1}{d}}\right).$$

In particular, if $G(\mathcal{S})$ has m edges, then it has a balanced separator of size $O_d(m^{1/d}n^{1-2/d})$.

Proof. For each $S \in \mathcal{S}$, pick an arbitrary point in the interior of S to be its center. Consider the partial order on $\mathcal{S}$ where $S \prec S'$ if S is contained in the interior of S'. Pick an arbitrary linear extension of this partial order. When we talk about the $S \in \mathcal{S}$ of maximal radius (or minimal radius) we mean maximal with respect to this linear extension. With these two definitions, we can proceed with the proof of Theorem 3.1 in the same way. The same randomized procedure produces a $(1 - c_d)$-balanced separator.

The main difference in the analysis is understanding $\mathbb{E}[|\mathcal{X}_3|]$. For each S, define $\tilde{r}(S)$ so that the $(d-1)$-volume of the intersection of the convex body bounded by S and the sphere S_{rand} is $\tau_{d-1}\tilde{r}(S)^{d-1}$. We claim that $\Pr[\tilde{r}(S) \le \tilde{r}] \le O_C(\tilde{r}/R)$.

Fix $\tilde{r}$. Change coordinates so that S_{rand} is centered at the origin. Write B for the convex body with boundary S. Let $r_{\min} = \min_{x\in B} \|x\|$ and $r_{\max} = \max_{x\in B} \|x\|$. Pick $p, q \in B$ with $\|p\| = r_{\min}$ and $\|q\| = r_{\max}$. Since $\|p - q\| \ge r_{\max} - r_{\min}$, we see that the smallest ball that contains B has radius at least $(r_{\max} - r_{\min})/2$. By the C-fat hypothesis, B contains some ball B' centered at a point o with radius $(r_{\max} - r_{\min})/2C$. Now B contains the convex hull of $\{p, q\} \cup B'$ which contains two cones with apexes p, q and with bases some disk of radius $(r_{\max} - r_{\min})/2C$ centered at o. This implies that for each $\epsilon > 0$, the body B contains a ball of radius $\epsilon(r_{\max} - r_{\min})/2C$ centered at a point of distance $r_{\min} + \epsilon(r_{\max} - r_{\min})$ from the origin and another ball of the same radius centered at a point of distance $r_{\max} - \epsilon(r_{\max} - r_{\min})$ from the origin. (See Figure 4.) In particular, if $\epsilon(r_{\max} - r_{\min})/2C \ge \tilde{r}$, then the intersection of B and the spheres of radius $r_{\min} + \epsilon(r_{\max} - r_{\min})$ and $r_{\max} - \epsilon(r_{\max} - r_{\min})$ both contain a spherical cap of radius at least $\epsilon(r_{\max} - r_{\min})/2C \ge \tilde{r}$, so these intersections have $(d-1)$-volume at least $\tau_{d-1}\tilde{r}^{d-1}$. The first inequality holds for $\epsilon \ge 2C\tilde{r}/(r_{\max} - r_{\min})$, so

$$\Pr[\tilde{r}(S) \le \tilde{r}] \le \frac{2(r_{\max} - r_{\min}) \cdot 2C\tilde{r}/(r_{\max} - r_{\min})}{R} = \frac{4C\tilde{r}}{R},$$

as desired.

This gives us the bound $\mathbb{E}[|\mathcal{X}_3|] \le \Sigma/4$ as in Claim 3.4. Then the bound $|\mathcal{X}_4| \le \Sigma/4$ also holds since we defined $\tilde{r}(S)$ so that the proof of Claim 3.5 still goes through. ◀

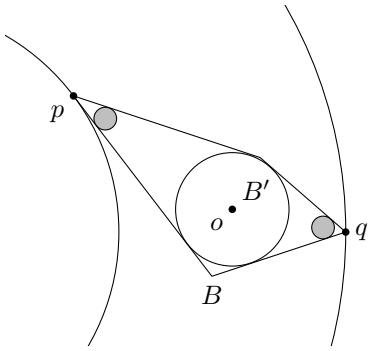


Figure 4 Bounding $\Pr[\tilde{r}(S) \leq \tilde{r}]$.

With these results, Corollary 3.8 also holds for fat convex bodies if the bodies are also semialgebraic sets of bounded description complexity.

We also note that both of the assumptions, convexity and fatness, are necessary. Indeed, for $d \geq 3$, every graph is the intersection graph of convex bodies in $\mathbb{R}^d$ and also every graph is the intersection graph of fat connected bodies in $\mathbb{R}^d$. The former fact is classical (see [22]); we prove the latter fact here.

▶ **Proposition 4.4.** *For $d \geq 3$, $\epsilon > 0$, and any graph G, there exists a collection of $(1+\epsilon)$-fat connected bodies in $\mathbb{R}^d$ whose intersection graph is G.*

Proof. Let B_k be the ball centered at the origin in $\mathbb{R}^d$ of radius $r_k = (2 + 2\epsilon^{-1})^k$. We will define X_k to be a $(1+\epsilon)$-fat connected body that is contained in B_k. The set X_k will be a "balloon" consisting of a large sphere in $\operatorname{int}(B_k) \setminus \operatorname{int}(B_{k-1})$ that is tangent to B_{k-1} at a point p_k and a "string" in B_{k-1}: a curve starting at $p_k \in \partial B_{k-1}$, passing through ∂B_0 at a point q_k, and then continuing into B_0. We will define these sets X_k so that they are disjoint in $\mathbb{R}^d \setminus \operatorname{int}(B_0)$. See Figure 5 for an illustration of this construction.

Suppose we have defined $B_0, \dots, B_{k-1}$ and $X_1, \dots, X_{k-1}$. Let p_k be a point on ∂B_{k-1}. Say B_{k-1} has radius r_{k-1}. Let Y_k be the ball of radius $\epsilon^{-1} r_{k-1}$, centered at $(1+\epsilon^{-1})p_k$. Then $B_{k-1} \cup Y_k$ is contained in the ball of radius $(1+\epsilon)\epsilon^{-1} r_{k-1}$ centered at $\epsilon^{-1} p_k$. Since $r_k = 2(1+\epsilon^{-1}) r_{k-1}$, we see that Y_k is contained in $\operatorname{int}(B_k)$. We will define X_k to be $Y_k \cup \gamma_k$ where γ_k is a curve starting from p_k which is contained in B_{k-1}. Whatever our choice of γ_k, we have ensured that X_k is $(1+\epsilon)$-fat and connected.

Pick q_k to be a point in $\partial B_0 \setminus \{q_1, \dots, q_{k-1}\}$. Let γ_k' be a curve from p_k to q_k whose interior is contained in

$$\operatorname{int}(B_{k-1}) \setminus \left(B_0 \cup \bigcup_{i=1}^{k-1} X_i \right).$$

If $k = 1$, we set $q_1 = p_1$ and let γ_1' have length 0. For $k > 1$, clearly there is enough space in dimension $d \geq 3$ to find such a γ_k'.

Finally, define $\gamma_k = \gamma_k' \cup \delta_k$ where δ_k is a curve starting from q_k whose interior is contained in $\operatorname{int}(B_0)$ such that δ_k and δ_ℓ intersect if and only if k, ℓ are adjacent in G. Again, in dimension $d \geq 3$, this is clearly possible. ◀

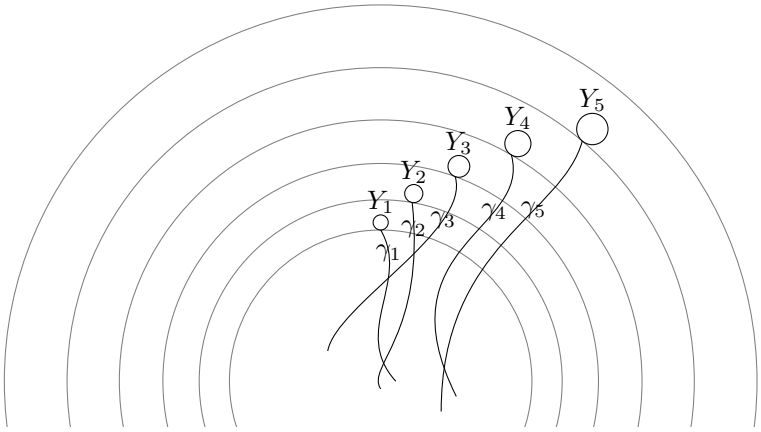


Figure 5 The construction in Proposition 4.4.

5 Algorithmic implementations

Suppose we are given $\mathcal{B}$ as a collection of centers and radii of balls. We give a randomized linear time algorithm that computes a small separator for $G(\mathcal{B})$.

▶ **Theorem 5.1.** *The separator for intersection graphs of balls given in Theorem 2.2 can be computed by a randomized linear time algorithm that succeeds with probability at least* $1/2$.

Proof. Write $\mathcal{P}$ for the centers of the balls of $\mathcal{B}$. Let r be the smallest radius such that there exists a ball B_0 of radius r that contains at least $9^{-d}n$ points of $\mathcal{P}$. Fix such a ball B_0. Then for any $p \in \mathcal{P} \cap B_0$, the ball of radius $2r$ centered at p contains B_0 and thus contains at least $9^{-d}n$ points of $\mathcal{P}$.

Our first goal is to efficiently find an approximation for this ball B_0. Fix a large constant C_d and sample $C_d \cdot 9^d$ points $p_1, p_2, \ldots \in \mathcal{P}$ uniformly at random. For each point p_i, compute the smallest r_i such that the ball of radius r_i centered at p_i contains $9^{-d}n$ points of $\mathcal{P}$. This can be computed in linear time for each p_i. Say we succeed if $\min_i r_i \leq 2r$. By the discussion above, we succeed if some $p_i \in \mathcal{P} \cap B_0$, which occurs with probability at least

$$1 - \left(1 - 9^{-d}\right)^{C_d \cdot 9^d} \geq 1 - e^{-C_d}.$$

Pick $1 \leq i \leq C_d \cdot 9^d$ such that r_i is minimal. As in the proof of Theorem 2.2, pick $r_{\text{rand}} \in [r_i, 2r_i]$ uniformly at random and let $\mathcal{X}$ be the set of balls $B \in \mathcal{B}$ which intersect the sphere of radius r_{rand} centered at p_i. This set can be computed in linear time. Now we still have

$$\mathbb{E}[|\mathcal{X}|] \leq 4 \left(\sum_{B \in \mathcal{B}} p(B)^{\frac{1}{d-1}} \right)^{1-\frac{1}{d}},$$

so by Markov's inequality,

$$\Pr\left[|\mathcal{X}| \leq 4C_d \left(\sum_{B \in \mathcal{B}} p(B)^{\frac{1}{d-1}} \right)^{1-\frac{1}{d}} \right] \geq 1 - C_d^{-1}.$$

Now the set $\mathcal{X}$ is a $(1-9^{-d})$-balanced separator for $G(\mathcal{B})$. To see this, note that the ball of radius r_i centered at p_i contains B_0 which contains $\lceil 9^{-d}n \rceil$ points of $\mathcal{P}$. Furthermore, the ball of radius $2r_i$ centered at p_i can be covered by at most 8^d translates of B_0, so it contains at most $8^d \lceil 9^{-d} \rceil n < (1-9^{-d})n$ points of $\mathcal{P}$.

Thus we have a randomized linear time algorithm which computes a small $(1-9^{-d})$-balanced separator for $G(\mathcal{B})$ with probability at least $1 - C_d^{-1}$. By Lemma 2.5, for C_d sufficiently large, we can iterate this algorithm $\lceil \log_{1-9^{-d}}(2/3) \rceil$ times to produce a $\frac{2}{3}$-balanced separator for $G(\mathcal{B})$ still in linear time with success probability at least $\frac{1}{2}$. ◀

Suppose we are given $\mathcal{S}$ as a list of centers and radii of the spheres. In quadratic time we can compute which pairs of spheres intersect and which pairs are contained in each other. From this we can run the algorithm described in Theorem 3.1 in quadratic time. The only potential difficulty is in selecting the minimal radius ball B_{in}, but this can be handled the same way is in Theorem 5.1.

▶ **Theorem 5.2.** *The separator for intersection graphs of spheres given in Theorem 3.1 can be computed by a randomized quadratic time algorithm that succeeds with probability at least* 1/2.

References

1 Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *J. Amer. Math. Soc.*, 3(4):801–808, 1990. doi:10.2307/1990903.

2 Béla Bollobás and Imre Leader. Edge-isoperimetric inequalities in the grid. *Combinatorica*, 11(4):299–314, 1991. doi:10.1007/BF01275667.

3 James Davies, Agelos Georgakopoulos, Meike Hatzel, and Rose McCarty. Strongly sublinear separators and bounded asymptotic dimension for sphere intersection graphs. In *41st International Symposium on Computational Geometry*, volume 332 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 36, 16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/lipics.socg.2025.36.

4 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for exponential-time-hypothesis-tight algorithms and lower bounds in geometric intersection graphs. *SIAM J. Comput.*, 49(6):1291–1331, 2020. doi:10.1137/20M1320870.

5 Zdeněk Dvořák and Serguei Norine. Small graph classes and bounded expansion. *J. Combin. Theory Ser. B*, 100(2):171–175, 2010. doi:10.1016/j.jctb.2009.06.001.

6 Jacob Fox and János Pach. Separator theorems and Turán-type results for planar intersection graphs. *Adv. Math.*, 219(3):1070–1080, 2008. doi:10.1016/j.aim.2008.06.002.

7 Jacob Fox and János Pach. A separator theorem for string graphs and its applications. *Combin. Probab. Comput.*, 19(3):371–390, 2010. doi:10.1017/S0963548309990459.

8 Jacob Fox and János Pach. Applications of a new separator theorem for string graphs. *Combin. Probab. Comput.*, 23(1):66–74, 2014. doi:10.1017/S0963548313000412.

9 John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5(3):391–407, 1984. doi:10.1016/0196-6774(84)90019-1.

10 Sariel Har-Peled. A simple proof of the existence of a planar separator, 2011. arXiv:1105.0103. arXiv:1105.0103.

11 Sariel Har-Peled and Kent Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. *SIAM J. Comput.*, 46(6):1712–1744, 2017. doi:10.1137/16M1079336.

12 Tuukka Korhonen and Daniel Lokshtanov. Induced-minor-free graphs: separator theorem, subexponential algorithms, and improved hardness of recognition. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5249–5275. SIAM, Philadelphia, PA, 2024. doi:10.1137/1.9781611977912.188.

13 James R. Lee. Separators in region intersection graphs. In *8th Innovations in Theoretical Computer Science Conference*, volume 67 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 1, 8. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.1.

14 Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 16(2):346–358, 1979. doi:10.1137/0716027.

15 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979. doi:10.1137/0136016.

16 Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980. doi:10.1137/0209046.

17 Jiří Matoušek. Near-optimal separators in string graphs. *Combin. Probab. Comput.*, 23(1):135–139, 2014. doi:10.1017/S0963548313000400.

18 Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, 1997. doi:10.1145/256292.256294.

19 Neil Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Combin. Theory Ser. B*, 92(2):325–357, 2004. doi:10.1016/j.jctb.2004.08.001.

20 W.D. Smith and N.C. Wormald. Geometric separator theorems and applications. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 232–243, 1998. doi:10.1109/SFCS.1998.743449.

21 Jonathan Tidor and Hung-Hsun Hans Yu. Multilevel polynomial partitioning and semialgebraic hypergraphs: regularity, Turán, and Zarankiewicz results, 2024. arXiv:2407.20221.

22 Heinrich Tietze. Über das Problem der Nachbargebiete im Raum. *Monatsh. Math. Phys.*, 16(1):211–216, 1905. doi:10.1007/BF01693778.

23 K. Wagner. Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.*, 114(1):570–590, 1937. doi:10.1007/BF01594196.

First-Order Logic and Twin-Width for Some Geometric Graphs

Colin Geniet
Discrete Mathematics Group, Institute for Basic Science (IBS), Daejeon, South Korea

Gunwoo Kim
School of Computing, KAIST, Daejeon, South Korea
Discrete Mathematics Group, Institute for Basic Science (IBS), Daejeon, South Korea

Lucas Meijer
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

For some geometric graph classes, tractability of testing first-order formulas is precisely characterised by the graph parameter *twin-width*. This was first proved for interval graphs among others in [BCKKLT, IPEC '22], where the equivalence is called *delineation*, and more generally holds for circle graphs, rooted directed path graphs, and H-graphs when H is a forest. Delineation is based on the key idea that geometric graphs often admit natural vertex orderings, allowing to use the very rich theory of twin-width for ordered graphs.

Answering two questions raised in their work, we prove that delineation holds for intersection graphs of non-degenerate axis-parallel unit segment graphs, but fails for visibility graphs of 1.5D terrains. We also prove delineation for intersection graphs of circular arcs.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Finite Model Theory; Theory of computation → Computational geometry; Mathematics of computing → Graph theory

Keywords and phrases Twin-width, axis-parallel unit segment graphs, circular arc graphs, terrain visibility graphs, first-order logic, model checking, FPT

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.51

Related Version *Full Version*: `https://arxiv.org/pdf/2512.21896`

Funding *Colin Geniet*: Supported by the Institute for Basic Science (IBS-R029-C1).
Gunwoo Kim: Supported by the Institute for Basic Science (IBS-R029-C1) and the National Research Foundation, Korea (RS-2025-00563533).
Lucas Meijer: Supported by the Netherlands Organisation for Scientific Research (NWO) under project no. VI.Vidi.213.150.

Acknowledgements The authors are extremely grateful to Eunjung Kim for suggesting the questions treated in this work and for very insightful discussions on these topics, as well as Sebastian Wiederrecht for stimulating discussions during the early stages of this project. We would also like to thank Édouard Bonnet for very helpful explanations regarding twin-width of segment graphs.

1 Introduction

The *first-order model checking* problem asks, given a graph G and a first-order (FO) formula ϕ, to test whether G satisfies ϕ. In general this is a difficult problem, known to be $\mathsf{AW}[*]$-hard [8], and subsuming problems such as finding in G an independent set, a dominating set, or any fixed induced subgraph. For a graph class $\mathcal{C}$, one may ask whether this FO model checking problem admits a *fixed parameter tractable* (FPT) algorithm, i.e. one running in time $f(|\phi|) \cdot |G|^{O(1)}$ for some (computable) function f.

© Colin Geniet, Gunwoo Kim, and Lucas Meijer;
licensed under Creative Commons License CC-BY 4.0

Classically, this problem has been studied in sparse graph classes, such as bounded degree graphs [17] and more generally nowhere dense classes [13], which admit such FPT algorithms. A more recent and incomparable development has come from the graph parameter *twin-width* [4]: if $\mathcal{C}$ is a class of bounded twin-width, and witnesses of this (so-called *contraction sequences*) can be efficiently computed, then FO model checking in $\mathcal{C}$ is FPT.

Since bounded twin-width and bounded degree are incomparable, a class with FPT model checking need not have bounded twin-width. Yet a number of remarkable cases exist in which twin-width is exactly the boundary of efficient FO model checking. A fundamental result of [3] is that for any hereditary (i.e. closed under induced subgraphs) class $\mathcal{C}$ of *ordered graphs* (i.e. graphs given with a total ordering of vertices, which formulas can query), model checking in $\mathcal{C}$ is FPT if and only if $\mathcal{C}$ has bounded twin-width[1]. By reducing to ordered graphs using a "nice" vertex ordering, similar results were proved for interval graphs and rooted directed path graphs in [1], where this equivalence is called *delineation*. It also holds for circle graphs [15], H-graphs for any fixed forest H [5], and tournaments [12]. We continue this line of delineation results for geometric intersection graphs:

▶ **Theorem 1.** *Model checking in $\mathcal{C}$ is FPT if and only if $\mathcal{C}$ has bounded twin-width (under the assumption $\mathsf{FPT} \neq \mathsf{AW}[*]$) in the following cases:*

1. *$\mathcal{C}$ is a hereditary subclass of intersection graphs of circular arcs, or*
2. *$\mathcal{C}$ is a hereditary subclass of intersection graphs of non-degenerate, axis-parallel unit segments, given by their geometric representation.*

In the axis-parallel unit segments (APUS) case, we require the geometric representation to be given, as computing it is NP-hard [16]. This is not an issue in circular arc graphs, where representations can be found in polynomial time [18]. We can also prove that relaxing any of the assumptions on non-degenerate APUS breaks delineation.

Finally, we show that delineation fails for visibility graphs of 1.5D-terrains (i.e. polygonal curves formed by an x-monotone sequence of points).

▶ **Theorem 2 (♣).** *There is a class $\mathcal{C}$ of 1.5D-terrain visibility graphs, whose hereditary closure allows FPT model checking, but which has unbounded twin-width.*

The questions of APUS graphs and 1.5D-terrain graphs were both raised in [1].

To further pursue this line of results, one may consider other shapes with unit size: whether unit disks graphs are delineated seems particularly interesting. The difficulty there might be to find the right notion of minimisation. One could also consider unit squares; we expect that our techniques can be used to prove delineation in the case of axis-aligned unit squares, and that delineation fails without the axis-aligned condition.

It may also make sense to consider other parameters than twin-width. Our non-delineated constructions of degenerate APUS graphs and terrain graphs use *merge-width* [10] to obtain FPT model checking. One can ask whether for these graphs, FPT model checking is equivalent to having bounded merge-width. Tackling this question would likely require characterising merge-width through some obstructions, a difficult open problem.

1.1 Proof overview

Let $\mathcal{D}$ be the graph class we are interested in, e.g. the class of all circular-arc graphs. Our goal is the following: for any hereditary[2] subclass $\mathcal{C} \subseteq \mathcal{D}$, there is an FPT model checking algorithm in $\mathcal{C}$ if and only if $\mathcal{C}$ has bounded twin-width. Inspired by [1, 15], we obtain such an equivalence by proving a dichotomy of the following form, for any subclass $\mathcal{C} \subseteq \mathcal{D}$:

[1] Under the assumption $\mathsf{FPT} \neq \mathsf{AW}[*]$, or equivalently that model checking for *all graphs* is not FPT.

[2] The hereditary assumption is necessary to avoid classes in which model checking is easy for trivial reasons, such as graphs of the form: a complex graph on n vertices, plus 2^n isolated vertices as padding.

- Either we can compute in polynomial time a contraction sequence for any $G \in \mathcal{C}$, witnessing that G has bounded twin-width. Then [4] gives FPT model checking in $\mathcal{C}$.
- Or $\mathcal{C}$ is *monadically independent*, which informally means that graphs in $\mathcal{C}$ can be used to encode arbitrary graphs through FO logic (this is formalised by *FO transductions*, see Section 2.4 for the definition). In that case, and if $\mathcal{C}$ additionally is hereditary, then FO model checking on the class of all graphs reduces to FO model checking on $\mathcal{C}$ [9], proving that the latter is hard for the parametrised complexity class $\mathsf{AW}[*]$ [8].

Let us point out that if $\mathcal{C}$ has bounded twin-width, then it is monadically dependent (the opposite of monadically independent). Thus a subclass $\mathcal{C} \subseteq \mathcal{D}$ has bounded twin-width if and only if it is monadically dependent; this is the precise definition of *delineation* in [1]. The above dichotomy is a slight strengthening of delineation, additionally requiring to be able to efficiently find contraction sequences. When $\mathcal{C}$ is hereditary, and assuming $\mathsf{FPT} \neq \mathsf{AW}[*]$, this strengthening gives the equivalence of $\mathcal{C}$ having bounded twin-width, $\mathcal{C}$ being monadically dependent, and $\mathcal{C}$ admitting an FPT model checking algorithm. Let us point out that it is a major conjecture in finite model theory that the last two points are always equivalent:

▶ **Conjecture 3** ([11, Conjecture 8.2]). *For any hereditary class $\mathcal{C}$, there is an FPT FO model checking algorithm for graphs in $\mathcal{C}$ if and only if $\mathcal{C}$ is monadically dependent.*

Our results confirm special cases of this, where twin-width explains the equivalence.

Coming back to the proof overview, let us explain how the previously stated dichotomy is achieved. The starting point is the characterisation of twin-width through obstructions in matrices [4]: given any 0–1 matrix M, one can efficiently find either a contraction sequence for M (witnessing small twin-width), or a *grid*. Given the geometric representation of the graph G, we construct some auxiliary matrix M_G. For instance, for circular arc graphs, we identify the circle with $\{1, \dots, n\}$ modulo n, and the matrix M has a 1 at position (i, j) whenever there is an arc clockwise from i to j. We then prove two facts:

- First, G can be reconstructed from M_G by a sequence of operations that preserve twin-width. Thus if M_G has bounded twin-width, then so does G. This is efficient: a contraction sequence for G can quickly be computed from one for M_G.
- Second, if M_G contains a sufficiently large grid, then G contains a complex structure called a *transversal pair*. It is known that if graphs in $\mathcal{C}$ contain arbitrarily large transversal pairs, then $\mathcal{C}$ is monadically independent [1].

Thus either one can find contraction sequences for the auxiliary matrices $\{M_G : G \in \mathcal{C}\}$, and thus also for $\mathcal{C}$, or these matrices have arbitrarily large grids, hence $\mathcal{C}$ contains arbitrarily large transversal pairs and is monadically independent. The case of APUS graphs requires one additional step: we construct *local* auxiliary matrices, corresponding to 1-by-1 squares in the segment representation. If any such matrix has a grid, then we find a transversal pair; if all local matrices have bounded twin-width, then so does the full graph.

2 Preliminaries

The proofs for results marked with (♣) can be found in the full version. For $n \in \mathbb{N}$, we denote by $[n]$ the interval of integers $\{1, \dots, n\}$.

2.1 Graphs and relational structures

For a graph $G = (V, E)$, we write $V(G) = V$ and $E(G) = E$ for its vertex and edge sets. The set of neighbours of $v \in V(G)$ is $N(v)$. Vertices x, y are *twins* if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$ (xy may or not be an edge). For $X \subseteq V$, $G[X]$ is the subgraph induced by X. When $<$ is a

linear ordering of $V(G)$, we denote by $\mathsf{Adj}(G, <)$ the adjacency matrix with rows and columns ordered by $<$. Similarly, if $<_V$ and $<_E$ are orderings of V and E, then $\mathsf{Inc}(G, <_V, <_E)$ is the incidence matrix with rows and columns ordered by $<_V$ and $<_E$.

More generally, a *binary relational structure* (or simply binary structure) $(V, E_1, \dots, E_\ell)$ consists of a vertex set V, and relations $E_i \subseteq V^2$. It can be viewed as a directed graph with ℓ kinds of edges, where there may be several edges of different kinds between two vertices (but not multiple edges of the same kind). Given a linear ordering $<$ of V, one defines the adjacency matrix of each relation E_i, denoted by $\mathsf{Adj}(E_i, <)$. An *ordered binary structure* is a binary structure in which one of the relations is a linear ordering of the vertex set, normally denoted by $<$. The structure is thus of the form $(V, E_1, \dots, E_\ell, <)$. A special case is *ordered graphs* $(V, E, <)$, where (V, E) is a graph and $<$ a linear ordering of V.

2.2 Grids

The following notion of grids comes from [14]. Given a point $p \in \mathbb{R}^2$, we write $\pi_x(p), \pi_y(p)$ for its x- and y-coordinates. A *t-grid* in the plane is a family of points $(p_{i,j})_{i,j\in[t]}$ satisfying the following for any $i, i', j, j' \in [t]$:

$$\text{if } i < i', \text{ then } \pi_x(p_{i,j}) < \pi_x(p_{i',j'}) \qquad \text{and} \qquad \text{if } j < j', \text{ then } \pi_y(p_{i,j}) < \pi_y(p_{i',j'}).$$

Observe that there are intervals $X_1 < \cdots < X_t$ and $Y_1 < \cdots < Y_t$ of $\mathbb{R}$ such that $p_{i,j} \in X_i \times Y_j$: take X_i to be the smallest interval containing $\pi_x(p_{i,j})$ for all i, and similarly with Y_j.

This extends to 0–1 matrices, interpreting a one at position (i, j) in the matrix M as a point with coordinates (i, j). We say that M contains a t-grid if the point set defined by the ones in M contains a t-grid in the previous sense. Then there are partitions into intervals $\mathcal{R} = \{X_1 < \cdots < X_t\}$ and $\mathcal{C} = \{Y_1 < \cdots < Y_t\}$ of the rows and columns as above. The ordering of rows and columns is crucial, which is why we make it explicit in $\mathsf{Adj}(G, <)$.

The Ramsey property for grids easily follows from the Ramsey theorem on bicliques:

▶ **Lemma 4** (Ramsey theorem for grids, ♣). *There is a function $R(t, k)$ such that if $\mathcal{P}$ is a point set containing an $R(t, k)$-grid, and $\lambda : \mathcal{P} \to [k]$ is any colouring, then $\mathcal{P}$ contains a t-grid consisting only of points of colour c for some $c \in [k]$.*

When the x- and y-coordinates of the grid are indexed by the same set (e.g. in an adjacency matrix), they can be assumed to come from disjoint intervals at a slight cost:

▶ **Lemma 5 (♣).** *Any $2t$-grid $\mathcal{P}$ contains a t-subgrid $\mathcal{P}' \subset \mathcal{P}$ such that the sets $X \coloneqq \pi_x(\mathcal{P}')$ and $Y \coloneqq \pi_y(\mathcal{P}')$ of x- and y-coordinates satisfy either $X < Y$ or $Y < X$.*

The following technical lemma relates grids in incidence and adjacency matrices.

▶ **Lemma 6 (♣).** *Let $E \subseteq V^2$ be a binary relation such that $\mathsf{Adj}(E, <_V)$ has no k-grid. Let $<_{\mathsf{lex}}$ be the lexicographic ordering of E, i.e. $(x, y) <_{\mathsf{lex}} (x', y')$ if either $x <_V x'$, or $x = x'$ and $y <_V y'$. Then $\mathsf{Inc}(E, <_V, <_{\mathsf{lex}})$ has no $4k$-grid.*

2.3 Twin-width

Twin-width [4] is defined by *contraction sequences*: a contraction sequence of width k is a witness that the twin-width $\mathsf{tww}(G)$ is at most k. We omit this definition, as we do not directly manipulate contraction sequences, instead relying on higher level results. It suffices to know that twin-width $\mathsf{tww}(G)$ is well-defined when G is a graph or a binary structure.

We will use the following observation, which is the origin of the name "twin-width".

▶ **Lemma 7** ([4]). *If x, y are twins in G, then* $\mathsf{tww}(G) = \mathsf{tww}(G - y)$.

The fundamental characterisation of twin-width is that it corresponds to ordered graphs whose adjacency matrix excludes some high-rank grid-like structures [3, Theorem 7]. This characterisation is furthermore efficient. It is simple to check that these high-rank grids contain our notion of grid, hence this implies the following.

▶ **Corollary 8** (of [4, Thm. 5.4] or [3, Thm. 7]). *There are functions f, g and an algorithm which, given $(V, E, <)$ an ordered graph whose matrix* $\mathsf{Adj}(E, <)$ *has no k-grid, computes a contraction sequence witnessing that* $\mathsf{tww}(V, E, <) \leq f(k)$ *in time $g(k) \cdot |V|^{O(1)}$.*

Ordered graphs of high twin-width have a Ramsey-like property:

▶ **Theorem 9** ([3]). *There is a function $R(k, \ell)$ such that if $(V, E, <)$ is an ordered graph with $E = E_1 \cup \cdots \cup E_\ell$, and* $\mathsf{tww}(V, E_i, <) \leq k$ *for all i, then* $\mathsf{tww}(V, E, <) \leq R(k, \ell)$.

Proof. This follows from the characterisation of bounded twin-width for ordered graphs by excluding so-called high-rank divisions [3, Theorems 7 and 23], and the Ramsey property for these high-rank divisions [3, Lemma 24]. ◀

It is easy to observe that the twin-width of a graph is the maximum twin-width of its connected components. We will use the following variant of this result for ordered graphs.

▶ **Lemma 10** (♣). *Let $G = (V, E, <)$ be an ordered graph, with V partitioned into intervals $A_1 < \cdots < A_m < B_1 < \cdots < B_m$, such that each edge is between A_i and B_i for some $i \in [m]$, and the subgraphs $(A_i \cup B_i, E, <)$ have twin-width at most k for all i. Then* $\mathsf{tww}(G) \leq 2k + 2$.

2.4 First-order logic

A *relational signature* is a set Σ of *relation symbols* $R \in \Sigma$, each with an arity $r \in \mathbb{N}$. In this work, we will only use *unary* and *binary* symbols, i.e. with arity 1 or 2. A *Σ-structure* S consists of a vertex set $V(S)$, and an interpretation of each symbol $R \in \Sigma$ as a relation $R(S) \subseteq (V(S))^r$ of appropriate arity. Thus a unary symbol U is interpreted as a vertex subset, and a binary symbol as a set of directed edges with loops. For instance, graphs are $\{E\}$-structures, and ordered graphs are $\{E, <\}$-structures (with $E, <$ binary symbols).

FO formulas ϕ over a signature Σ consist of quantifiers over vertices $\forall x, \phi$, $\exists x, \phi$, boolean operators $\vee, \wedge, \neg$, and predicates $R(x_1, \ldots, x_r)$ testing whether $(x_1, \ldots, x_r)$ is a tuple in the interpretation of R, as well as the equality predicate $x = y$. For example,

$$\phi(x, y) = (x = y) \vee E(x, y) \vee \exists z.\ (E(x, z) \wedge E(z, y))$$

is a formula over the signature $\{E\}$ (for graphs) expressing that x, y are at distance at most two. When the formula ϕ has free variables, they are written as arguments of ϕ as above. A formula without free variables is called a *sentence*. If $\phi(x_1, \ldots, x_r)$ is a formula over Σ, S is a Σ-structure, and $v_1, \ldots, v_r \in V(S)$ are vertices, then $S \models \phi(v_1, \ldots, v_r)$ denotes that ϕ is *satisfied* by S, with the variables $x_1, \ldots, x_r$ interpreted as $v_1, \ldots, v_r$ respectively.

The FPT model checking algorithm for twin-width shows the following.

▶ **Theorem 11** ([4, Theorem 1]). *Given a graph G, a contraction sequence of width k for G, and a sentence ϕ, one can test whether $G \models \phi$ in time $f(k, |\phi|) \cdot |V(G)|$.*

Given two signatures Σ, Γ, an *FO interpretation* Φ from Σ to Γ is a map, defined in FO logic, from Σ-structures to Γ-structures. Precisely, Φ is described by giving
1. for each symbol $R \in \Gamma$ of arity r, an FO formula $\phi_R(x_1, \dots, x_r)$ over Σ, and
2. one last FO formula $\phi_V(x)$ again over Σ.

Given a Σ-structure S, its image $\Phi(S)$ is the relational structure where the vertex set is restricted to vertices satisfying ϕ_V, i.e.

$$V(\Phi(S)) = \{x \in V(S) \ : \ S \models \phi_V(x)\},$$

and for each symbol $R \in \Gamma$, the relation set $R(\Phi(S))$ is described by ϕ_R, i.e.

$$R(\Phi(S)) = \{(x_1, \dots, x_k) \in (V(\Phi(S)))^r \ : \ S \models \phi_R(x_1, \dots, x_r)\}.$$

Transductions are a non-deterministic generalisation of interpretations, i.e., a Σ-to-Γ transduction outputs a set of Γ-structures, rather than a single one. Let Σ be a signature, and $U_1, \dots, U_k$ be k unary symbols, disjoint from Σ. The k-colouring is the one-to-many operation which maps a Σ-structure S to all possible extensions of S as $(\Sigma \uplus \{U_1, \dots, U_k\})$-structures S^+, meaning that $V(S) = V(S^+)$ and $R(S) = R(S^+)$ for any $R \in \Sigma$, while the $U_i(S^+)$ are chosen as arbitrary subsets of $V(S^+)$. An *FO transduction* is the composition of a k-colouring (for fixed k), followed by an FO interpretation.

It is folklore that interpretations and transductions can be composed.

▶ **Lemma 12.** *If Φ, Ψ are FO transductions (resp. interpretations) from Σ to Γ and Γ to Δ respectively, then the composition $\Psi \circ \Phi$ is an FO transduction (interpretation) from Σ to Δ.*

Transductions preserve bounded twin-width, and this can be made efficient.

▶ **Theorem 13** ([4, Theorem 8.1])**.** *For any FO transduction Φ, there is a function f such that for any binary structure S and $T \in \Phi(S)$, $\mathsf{tww}(T) \leq f(\mathsf{tww}(S))$.*

Furthermore, given a contraction sequence of width k for S, and the colouring of S used to obtain T through Φ, one can compute in time $g(k) \cdot |S|^{O(1)}$ a contraction sequence of width $f(k)$ for T for some function g depending on Φ.

A graph class $\mathcal{C}$ is called *monadically independent* if it FO transduces the class of all graphs, i.e. if there exists a transduction Φ such that $\Phi(\mathcal{C})$ contains all graphs. If there is no such transduction, then $\mathcal{C}$ is called *monadically dependent*. Theorem 13 implies that bounded twin-width classes are monadically dependent.

▶ **Theorem 14** ([9])**.** *If $\mathcal{C}$ is a hereditary monadically independent class of graphs, then FO model checking in $\mathcal{C}$ is $\mathsf{AW}[*]$-hard.*

To prove that a class $\mathcal{C}$ is monadically independent, we will use the next construction, proposed in [1]. Our definition differs slightly, but the idea and purpose are the same.

▶ **Definition 15.** *A* transversal pair T_k *consists of three sets $A = \{a_i : i \in [k]\}$, $B = \{b_{i,j} : i, j \in [k]\}$ and $C = \{c_j : j \in [k]\}$ with edges $a_i b_{i',j'}$ if and only if $i \leq i'$, and $b_{i,j} c_{j'}$ if and only if $j \leq j'$ (see Figure 1). Edges within A, B, C or between A and C are unconstrained.*

▶ **Lemma 16 (♣).** *Let $\mathcal{C}$ be a graph class such that for any k, some $G \in \mathcal{C}$ contains a transversal pair T_k. Then $\mathcal{C}$ is monadically independent.*

2.5 Geometric graphs

Given a set family $\mathcal{F}$, the *intersection graph* $G_{\mathcal{F}}$ is defined with $\mathcal{F}$ as its vertex set, and an edge between $A, B \in \mathcal{F}$ if and only if $A \cap B \neq \varnothing$. The family $\mathcal{F}$ is a *representation* of $G_{\mathcal{F}}$.

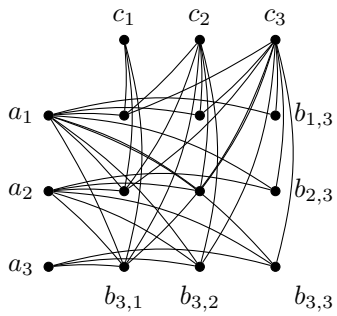


Figure 1 A transversal pair T_3.

Circular arc graphs. A *circular arc graph* is an intersection graph of arcs around a circle. We work with discrete representations in $[n] \coloneqq \{1, ..., n\}$ modulo n: a (discrete) *circular arc* is an interval $[i, j]$ modulo n; explicitly, $[i, j] \coloneqq \{k : i \leq k \leq j\}$ when $i \leq j$, and $[i, j] \coloneqq \{k : k \geq i \text{ or } k \leq j\}$ when $i > j$. Any circular arc representation $\mathcal{F}$ of G can be turned into a discrete circular arcs representation as above for $n = 2|V(G)|$, in time polynomial in the size of the encoding of $\mathcal{F}$. We often identify the vertices of G with the arcs of $\mathcal{F}$. When $I = [i, j]$ is an arc, $\mathsf{left}(I) \coloneqq i$ and $\mathsf{right}(I) \coloneqq j$ denote its endpoints.

▶ **Theorem 17** ([18]). *Given a graph G, one can compute a circular arc representation of G or detect that it is not a circular arc graph in polynomial time.*

Axis-parallel unit segment graphs. An axis-parallel unit segment (APUS) is a segment of length 1 in the plane that is horizontal or vertical. We denote by $\mathsf{Hor}(x, y)$ the horizontal segment from (x, y) to $(x + 1, y)$, and by $\mathsf{Ver}(x, y)$ the vertical one from (x, y) to $(x, y + 1)$. An APUS graph is the intersection graph of a family of APUS. We again work with discrete representations: Any APUS graph G has a representation $\mathcal{F}$ using coordinates that are non-negative multiples of $\eta(\mathcal{F}) = \frac{1}{|\mathcal{F}|}$. This follows from a similar result on unit interval graphs [7], applied to vertical and horizontal coordinates independently.

A family $\mathcal{F}$ of APUS is *non-degenerate* if no two horizontal (resp. two vertical) segments of $\mathcal{F}$ intersect. Intersection graphs of non-degenerate APUS families are called non-degenerate APUS graphs. This is equivalent to G being the intersection graph of an arbitrary family $\mathcal{F}$ of APUS, ignoring intersections between horizontal (vertical) segments (i.e. G has no edge for such intersections). Non-degenerate APUS graphs have a natural bipartition into horizontal and vertical segments. We use the second definition of non-degenerate APUS graphs, that is, degenerate intersections can exist, but are ignored in intersection graphs. The reason is that we will use APUS representations with minimized coordinates (see Section 4).

3 Delineation of circular arc graphs

In this section, we prove Theorem 1 for circular arc graphs.

Given a family $\mathcal{F}$ of circular arcs in $[n]$, we define its *endpoint matrix* $M_{\mathcal{F}}$, where rows and columns are $[n]$, and there is a 1 at position (i, j) if and only if $[i, j] \in \mathcal{F}$.

▶ **Lemma 18.** *There are functions f, g such that if $M_{\mathcal{F}}$ has no k-grid, then one can compute a contraction sequence witnessing that $\mathsf{tww}(G_{\mathcal{F}}) \leq f(k)$ in FPT time $g(k) \cdot |\mathcal{F}|^{O(1)}$.*

Proof. In a first time, let us assume that $\mathcal{F}$ does not contain multiple identical circular arcs.

We identify intervals $[x, y] \in \mathcal{F}$ with pairs (x, y), so that $\mathcal{F}$ describes a binary relation on $[n]$. Its adjacency matrix $\mathsf{Adj}(\mathcal{F}, <)$ with the natural ordering of $[n]$ is exactly $M_{\mathcal{F}}$. It has no k-grid, hence by Lemma 6, the incidence matrix $\mathsf{Inc}(\mathcal{F}, <, <_{\mathsf{lex}})$ has no $4k$-grid, where $<_{\mathsf{lex}}$ is the lexicographic ordering of $[n]^2$.

Construct the following ordered graph $H = (V, E, <_V)$. The vertex set is $V \coloneqq [n] \uplus \mathcal{F}$, ordered with $[n]$ before $\mathcal{F}$, with the natural ordering inside $[n]$, and the lexicographic ordering inside $\mathcal{F}$. In E, each interval $[x, y] \in \mathcal{F}$ is connected to its two endpoints. Thus $\mathsf{Adj}(H, <_V)$ consists of $\mathsf{Inc}(\mathcal{F}, <, <_{\mathsf{lex}})$ and its transpose arranged as diagonal blocks. Since $\mathsf{Inc}(\mathcal{F}, <, <_{\mathsf{lex}})$ has no $4k$-grid, one may check that this implies that $\mathsf{Adj}(H, <_V)$ has no $(4k + 1)$-grid. Using Corollary 8, one can compute in FPT time a contraction sequence of bounded width for H.

To conclude using Theorem 13, we build an FO transduction Φ such that $G_{\mathcal{F}} \in \Phi(H)$:

1. Firstly, add colours to V to distinguish $[n]$ and $\mathcal{F}$, and also to indicate for each interval $[x, y] \in \mathcal{F}$ whether or not $x \leq y$ (this is 3 colours in total).
2. Using these colours, we can write a formula $\phi(v, e)$ testing if $v \in [n]$ belongs to the interval $e \in \mathcal{F}$. Indeed, let $x, y \in [n]$ be the neighbours of e, with $x < y$. Then the interval e is either $[x, y]$ or $[y, x]$, indicated by the colour given to e at step 1. It suffices for $\phi(v, e)$ to check that $x \leq v \leq y$ in the first case, and that $v \leq y \vee x \leq v$ in the second.
   As an extreme case, one could have $e = [x, x]$, which is adjacent to x only. Then x is the only point contained in e. All of the above is easily expressed by a first-order formula.
3. Now, one can test if $e, f \in \mathcal{F}$ intersect with the formula $\psi(e, f) = \exists v.\ \phi(e, v) \wedge \phi(f, v)$.
4. Finally, vertices of $[n]$ are deleted, keeping only $\mathcal{F}$ and the edges defined by ψ.

This is a transduction which, for the choice of colours specified at step 1, yields $G_{\mathcal{F}}$ from H. By Theorem 13, we obtain a contraction sequence for $G_{\mathcal{F}}$ from the one for H in FPT time.

It remains to consider the case where $\mathcal{F}$ has multiple identical arcs. Let $\mathcal{F}'$ be $\mathcal{F}$ with these duplicates removed. Then $M_{\mathcal{F}'} = M_{\mathcal{F}}$, and $G_{\mathcal{F}'}$ is $G_{\mathcal{F}}$ with some twins removed, hence by Lemma 7 $\mathsf{tww}(G_{\mathcal{F}}) = \mathsf{tww}(G_{\mathcal{F}'})$. We conclude using the results for $\mathcal{F}'$. ◀

Say a circular arc representation $\mathcal{F}$ is *minimized* if its circular arcs are inclusion-wise minimal while representing $G_{\mathcal{F}}$, i.e. for any $I \in \mathcal{F}$, removing either endpoint of I would cause it to no longer intersect some other arc $I' \in \mathcal{F}$. A circular arc representation $\mathcal{F}$ can easily be minimized in time polynomial in the size of the encoding of $\mathcal{F}$.

▶ **Lemma 19.** *If $\mathcal{F}$ is a minimized circular arc representation and $I = [i, j] \in \mathcal{F}$ with $i \neq j$, then there exist $I_L, I_R \in \mathcal{F}$ such that $I \cap I_L = \{i\}$ and $I \cap I_R = \{j\}$.*

Proof. If I violates the condition, then it can be shortened by deleting either i or j. ◀

▶ **Lemma 20.** *Let k be a positive integer. Consider a minimized circular arc representation $\mathcal{F}$ of a graph $G_{\mathcal{F}}$. If $M_{\mathcal{F}}$ contains a $(4k + 2)$-grid, then $G_{\mathcal{F}}$ contains a transversal pair T_k.*

Proof. Consider a $(4k + 2)$-grid in $M_{\mathcal{F}}$. It consists of intervals $I_{i,j}$ with $\mathsf{left}(I_{i,j}) < \mathsf{left}(I_{i',j'})$ whenever $i < i'$, and $\mathsf{right}(I_{i,j}) < \mathsf{right}(I_{i',j'})$ whenever $j < j'$. By Lemma 5, a $(2k + 1)$-grid can be extracted so that the x- and y-coordinates belong to disjoint intervals. Thus we assume that $i, j \in [2k + 1]$, and that $\mathsf{left}(I_{i,j}) < \mathsf{right}(I_{i',j'})$ for all i, i', j, j', the case $\mathsf{left}(I_{i,j}) > \mathsf{right}(I_{i',j'})$ being similar.[3]

Let us start constructing the transversal pair T_k. Its central vertices $b_{i,j} \coloneqq I_{2i,2j}$ are obtained by picking every other row and column of the grid. For the left vertices a_i, we first extract auxiliary intervals $x_i \coloneqq I_{2i+1,2k+1}$ for $i \in [k]$. We claim that for all $i, i', j' \in [k]$

[3] This does not reduce the problem to an interval graph, as we will still use arcs other than the $I_{i,j}$s.

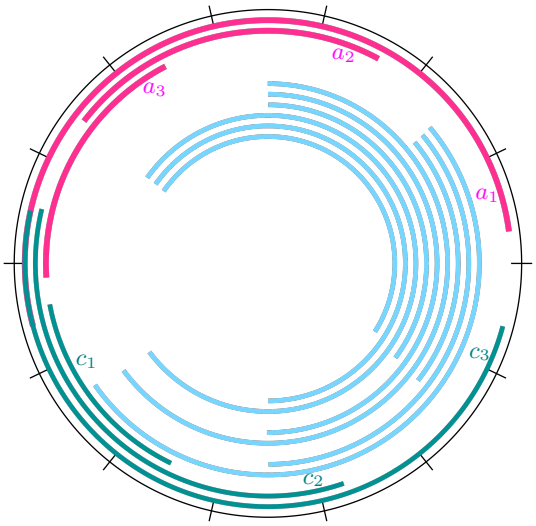


Figure 2 A circular arc representation of a transversal pair T_3 with the tripartition (A, B, C). Magenta circular arcs represent vertices in A, cyan circular arcs in B, and teal circular arcs in C.

$$\text{if } i \geq i', \text{ then } \mathsf{left}(x_i) \in b_{i',j'}, \text{ and} \tag{1}$$

$$\text{if } i < i', \text{ then } b_{i',j'} \subseteq x_i \setminus \{\mathsf{left}(x_i)\}. \tag{2}$$

Indeed, if $i \geq i'$, then $2i + 1 > 2i'$, which gives

$$\mathsf{left}(b_{i',j'}) = \mathsf{left}(I_{2i',2j'}) < \mathsf{left}(I_{2i+1,2k+1}) = \mathsf{left}(x_i). \tag{3}$$

In addition, since all left endpoints are smaller than all right endpoints for intervals of the grid, we also have $\mathsf{left}(x_i) < \mathsf{right}(b_{i',j'})$. Thus $b_{i',j'}$ contains $\mathsf{left}(x_i)$, proving (1). On the other hand, if $i < i'$, then we have $2i + 1 < 2i'$ and $2j' < 2k + 1$, which gives

$$b_{i',j'} = I_{2i',2j'} \subset I_{2i+1,2k+1} = x_i, \tag{4}$$

the inclusion being strict at both endpoints, which implies (2).

Using the minimality of $\mathcal{F}$ and Lemma 19, there is an interval $a_i \in \mathcal{F}$ such that $x_i \cap a_i = \{\mathsf{left}(x_i)\}$. It follows from (1) and (2) that a_i and $b_{i',j'}$ intersect if and only if $i \geq i'$. Symmetrically, we define $y_j = I_{1,2j-1}$, use Lemma 19 to find c_j such that $y_j \cap c_j = \{\mathsf{right}(y_j)\}$, and obtain that c_j and $b_{i',j'}$ intersect if and only if $j \leq j'$. Up to reversing the order of indices, the former description of $a_i, b_{i,j}, c_j$ gives a transversal pair. ◀

Our main theorem for circular arc graphs directly follows from Lemmas 18 and 20.

▶ **Theorem 21.** *Let $\mathcal{C}$ be a subclass of circular arc graphs. Then $\mathcal{C}$ has bounded twin-width if and only if it is monadically dependent. Further, when $\mathcal{C}$ is hereditary, and assuming* FPT $\neq$ AW[$*$]*, these are also equivalent to FO model checking in $\mathcal{C}$ being FPT.*

Proof. For each $G \in \mathcal{C}$, call $\mathcal{F}_G$ the circular arc representation computed by Theorem 17, after minimizing it. Consider the endpoint matrices $\mathcal{M} = \{M_{\mathcal{F}_G} : G \in \mathcal{C}\}$. If matrices in $\mathcal{M}$ have no k-grid for some $k \in \mathbb{N}$, then by Lemma 18 contraction sequences of width $f(k)$ can be computed in FPT time for graphs in $\mathcal{C}$. Thus $\mathcal{C}$ has bounded twin-width (and thus is monadically dependent), and model checking in $\mathcal{C}$ is FPT by Theorem 11.

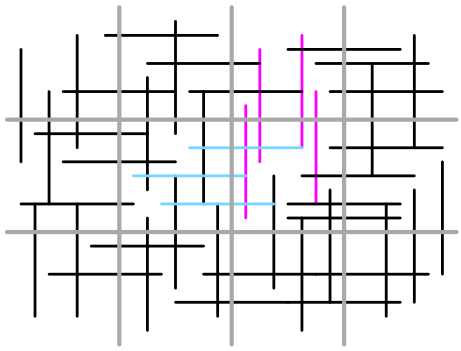


Figure 3 An instance being split along unit squares (drawn in gray). The cyan segments are the set $\mathcal{H}_{s-1,t}$, while the magenta segments are the set $\mathcal{V}_{s,t}$. Together, they form $\mathcal{F}_{s,t,\ulcorner}$.

If however matrices in $\mathcal{M}$ have arbitrarily large grids, then by Lemma 20, $\mathcal{C}$ contains arbitrarily large transversal pairs. Thus $\mathcal{C}$ is monadically independent by Lemma 16. If $\mathcal{C}$ is hereditary, this implies by Theorem 14 that model checking in $\mathcal{C}$ is AW[$*$]-hard. ◀

4 Delineation of axis-parallel unit segment graphs

This section proves Theorem 1 for APUS graphs. Recall that we work with non-degenerate APUS graphs: intersections between two horizontal (resp. vertical) segments are ignored.

As in the circular arc case, we associate the *endpoint matrix* $M_{\mathcal{F}}$ to a family $\mathcal{F}$ of APUS. The columns (resp. rows) are the x- (y-) coordinates of all endpoints of segments in $M_{\mathcal{F}}$, and the entry at position (x, y) is 1 if and only if $\mathsf{Hor}(x, y) \in \mathcal{F}$ or $\mathsf{Ver}(x, y) \in \mathcal{F}$. A variant of Lemma 18 shows that when $M_{\mathcal{F}}$ has no k-grid, then $G_{\mathcal{F}}$ has bounded twin-width. However, the converse fails: even with appropriate minimality conditions on $\mathcal{F}$, a grid in $M_{\mathcal{F}}$ does not imply that $G_{\mathcal{F}}$ has unbounded twin-width. This e.g. fails if the grid is formed by segments that are pairwise at distance more than 1 from each other. The solution, inspired by [1, Lemma 56], is to partition the APUS graph into regions corresponding to unit squares, and look for a grid in each region independently.

4.1 Splitting along unit squares

In an APUS family $\mathcal{F}$, for $s, t \in \mathbb{Z}$, denote by $\mathcal{H}_{s,t}$ the subset of segments $\mathsf{Hor}(x, y)$ where $x \in [s, s+1)$ and $y \in [t, t+1)$, and define $\mathcal{V}_{s,t}$ similarly for vertical segments. Clearly the sets $\mathcal{H}_{s,t}, \mathcal{V}_{s,t}$ for $s, t \in \mathbb{Z}$ partition $\mathcal{F}$. Also, observe that segments in $\mathcal{H}_{s,t}$ and $\mathcal{V}_{s',t'}$ can intersect only if $s' \in \{s, s+1\}$ and $t' \in \{t-1, t\}$. Let $\mathcal{F}_{s,t,\urcorner}$ denote the set $\mathcal{H}_{s,t} \cup \mathcal{V}_{s,t}$, and $E_{s,t,\urcorner}$ the edges of $G_{\mathcal{F}}$ contained in $\mathcal{F}_{s,t,\urcorner}$ that come from segments intersecting in the unit square $[s, s+1) \times [t, t+1)$, for segments coming from the right and top sides of this square (hence $\urcorner$ in the notation). Similarly, define $\mathcal{F}_{s,t,\ulcorner} = \mathcal{H}_{s-1,t} \cup \mathcal{V}_{s,t}$, $\mathcal{F}_{s,t,\lrcorner} = \mathcal{H}_{s,t} \cup \mathcal{V}_{s,t-1}$, and $\mathcal{F}_{s,t,\llcorner} = \mathcal{H}_{s-1,t} \cup \mathcal{V}_{s,t-1}$, as well as $E_{s,t,\ulcorner}, E_{s,t,\lrcorner}, E_{s,t,\llcorner}$ the corresponding edge sets. The sets $E_{s,t,d}$ for $s, t \in \mathbb{Z}$ and $d \in \{\urcorner, \ulcorner, \lrcorner, \llcorner\}$ partition the edge set of $G_{\mathcal{F}}$. We work on each $E_{s,t,d}$ independently with the techniques of Section 3, and show that when all have bounded twin-width, then so does $G_{\mathcal{F}}$. Let us start with the latter.

In general, edge partitions do not preserve bounded twin-width, but they do for ordered graphs (Theorem 9). To use this result, we define an ordering $<_{\mathsf{seg}}$ of segments.

A natural attempt is to take the lexicographic ordering $<_{\mathsf{lex}}$ of segments, i.e. $\mathsf{Hor}(x,y) <_{\mathsf{lex}} \mathsf{Hor}(x',y')$ if and only if $x < x'$, or $x = x'$ and $y < y'$, and similarly for vertical segments. However this works poorly with twin-width: consider a family of segments $\mathcal{F} = \{h_1, \dots, h_n, v_1, \dots, v_n\}$ such that h_i, v_i intersect close to $(0, 2i)$. In particular, h_i, v_j do not intersect for $i \neq j$. By slightly tweaking their x-coordinates, any permutation of $v_1, \dots, v_n$ can be realised as the lexicographic ordering $<_{\mathsf{lex}}$. Thus even for these very simple segment families, the ordered intersection graph $(\mathcal{F}, E, <_{\mathsf{lex}})$ can be an arbitrary ordered matching, which are known to have unbounded twin-width (see e.g. [3]).

The solution is to have two layers of lexicographic ordering, at local and global scales. That is, we first order the sets $\mathcal{H}_{s,t}$ lexicographically by (s,t), and then inside each $\mathcal{H}_{s,t}$ order the segments $\mathsf{Hor}(x,y)$ lexicographically as in $<_{\mathsf{lex}}$. The same ordering applies to vertical segments. Finally, all horizontal segments are placed before all vertical ones. Crucially, in $<_{\mathsf{seg}}$, and unlike $<_{\mathsf{lex}}$, each set $\mathcal{H}_{s,t}$ or $\mathcal{V}_{s,t}$ is an interval.

For an APUS family $\mathcal{F}$ with intersection graph G, denote by $G^{\mathsf{seg}} := (\mathcal{F}, E, <_{\mathsf{seg}})$ the same intersection graph ordered by $<_{\mathsf{seg}}$. Further, for $s,t \in \mathbb{Z}$ and $d \in \{\urcorner, \ulcorner, \lrcorner, \llcorner\}$, let $G^{\mathsf{seg}}_{s,t,d} = (\mathcal{F}_{s,t,d}, E_{s,t,d}, <_{\mathsf{seg}})$ be the restriction to $\mathcal{F}_{s,t,d}$.

▶ **Lemma 22.** *If* $\mathsf{tww}(G^{\mathsf{seg}}_{s,t,d}) \leq k$ *for all* s,t,d*, then* $\mathsf{tww}(G^{\mathsf{seg}}) \leq f(k)$ *for some function* f*.*

Proof. Group the edge sets $E_{s,t,d}$ into 4 classes according to the direction d, i.e. define $\mathcal{E}_d := \bigcup_{s,t} E_{s,t,d}$. The sets $\mathcal{E}_d$ for $d \in \{\urcorner, \ulcorner, \lrcorner, \llcorner\}$ are a partition of the edges, hence by Theorem 9, it suffices to prove that each $(\mathcal{F}, \mathcal{E}_d, <_{\mathsf{seg}})$ has bounded twin-width.

We consider $\mathcal{E}_{\urcorner}$; the other cases are similar. In $\mathcal{E}_{\urcorner}$, vertices of $\mathcal{H}_{s,t}$ are only adjacent to those of $\mathcal{V}_{s,t}$. Also, each $\mathcal{H}_{s,t}$ and $\mathcal{V}_{s,t}$ is an interval of $<_{\mathsf{seg}}$. The intervals $\mathcal{H}_{s,t}$ are ordered lexicographically by (s,t) between themselves, and the corresponding lexicographic ordering is used for the $\mathcal{V}_{s,t}$. Finally, all $\mathcal{H}_{s,t}$ come before all $\mathcal{V}_{s',t'}$. This fits all the requirements of Lemma 10, proving that $(\mathcal{F}, \mathcal{E}_d, <_{\mathsf{seg}})$ has twin-width at most $2k+2$. ◀

4.2 APUS graphs within a unit square

Let us now focus on each $G^{\mathsf{seg}}_{s,t,d}$, to find either bounded twin-width or a transversal pair.

Here, it must be mentioned that the graph $G^{\mathsf{seg}}_{s,t,d}$ is the complement of a circular arc graph, see Figure 4. It is thus not surprising that the techniques of Section 3 can be adapted. The results however cannot be applied as a blackbox: to construct transversal pairs when the matrix $M_{\mathcal{F}_{s,t,d}}$ has a large grid, we may also need segments outside $\mathcal{F}_{s,t,d}$.

We first show that if $M_{\mathcal{F}_{s,t,d}}$ has no k-grid, its intersection graph has bounded twin-width. By symmetry, it is enough to prove the result for $\mathcal{F}_{0,0,\urcorner}$. The only difference with Lemma 18 is that we now care about the ordered graph, with ordering $<_{\mathsf{seg}}$.

▶ **Lemma 23 (♣).** *Let* $\mathcal{F}$ *be a family of APUS consisting of* $\mathsf{Hor}(x,y)$ *or* $\mathsf{Ver}(x,y)$ *with* $x,y \in [0,1)$*. If the matrix* $M_{\mathcal{F}}$ *has no* k*-grid, then* $\mathsf{tww}(G^{\mathsf{seg}}_{\mathcal{F}}) \leq f(k)$ *for some function* f*.*

We now show that when $M_{\mathcal{F}_{s,t,d}}$ does contain a grid, then $G_{\mathcal{F}}$ contains a transversal pair. As in Section 3, this requires some minimality hypothesis. We assume discrete coordinates as in Section 2.5, i.e. in an APUS family $\mathcal{F}$, the coordinates of segments are non-negative multiples of some $\eta(\mathcal{F}) > 0$. Subject to this condition, call $\mathcal{F}$ *minimized* if its segments cannot be pushed towards the bottom-left while preserving the intersection graph. Formally, for a fixed $\eta = \eta(\mathcal{F})$, we say that $\mathcal{F}$ is minimized if there is no other family $\mathcal{F}'$ with the same $\eta = \eta(\mathcal{F}')$ such that there is a bijection $\phi : \mathcal{F} \to \mathcal{F}'$ that preserves the vertical/horizontal orientation and induces an isomorphism of the intersection graphs, and such that the endpoints of $\phi(s)$ are coordinate-wise no larger than that of s.

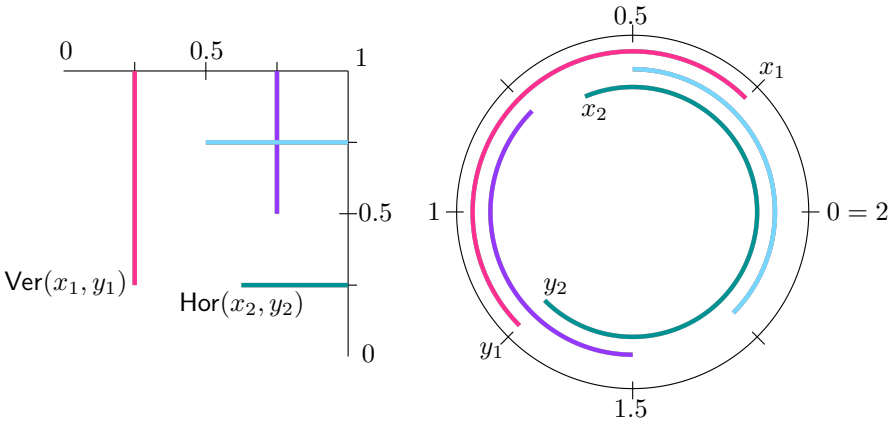


Figure 4 Constructing $G_{0,0,\urcorner}$ as the complement of a circular arc graph: map the segment $\mathsf{Hor}(x, y)$ to the arc $(1 + y, x)$, and $\mathsf{Ver}(x, y)$ to $(x, 1 + y)$ (where the arcs are in $[0, 2)$ modulo 2).

▶ **Lemma 24.** *In a minimized APUS family $\mathcal{F}$, for any horizontal $\mathsf{Hor}(x, y) \in \mathcal{F}$, there are:*
1. *a vertical segment $\mathsf{Ver}(x - \eta, y')$ or $\mathsf{Ver}(x + 1, y')$ with $y' \leq y \leq y' + 1$, unless $x = 0$, and*
2. *a vertical segment $\mathsf{Ver}(x', y - 1 - \eta)$ or $\mathsf{Ver}(x', y)$ with $x \leq x' \leq x + 1$, unless $y = 0$,*

and similarly when flipping the roles of vertical and horizontal, and of x- and y-coordinates.

Proof. If that is not the case, $\mathsf{Hor}(x, y)$ can be moved to the left (in case 1) or the bottom (in case 2) by η without changing the intersection graph, contradicting minimality. ◀

▶ **Lemma 25 (♣).** *Consider a minimized APUS family $\mathcal{F}$, and its restriction $\mathcal{F}_{s,t,d}$ for some $s, t \in \mathbb{Z}$ and $d \in \{\urcorner, \ulcorner, \lrcorner, \llcorner\}$. There is a function $f : \mathbb{N} \to \mathbb{N}$ such that if $M_{\mathcal{F}_{s,t,d}}$ has an $f(k)$-grid, then $G_{\mathcal{F}}$ contains a transversal pair T_k.*

Proof sketch. Consider an $f(k)$-grid $\mathcal{P}$ (for f to be determined) in $M_{\mathcal{F}_{s,t,d}}$, consisting of points $(p_{i,j})_{i,j\in[f(k)]}$ as described in Section 2.2. For each $i, j \in [f(k)]$, $\mathcal{F}$ contains a horizontal segment with $p_{i,j}$ as left endpoint or a vertical segment with $p_{i,j}$ as bottom endpoint.

Using Ramsey's theorem (Lemma 4), when $f(k)$ is chosen large enough, we can find a $2k$-subgrid $\mathcal{P}'$ in which all segments have the same orientation, and their minimality is ensured in the same manner (in the sense of cases (1) or (2) of Lemma 24). Without loss of generality, we assume that segments of $\mathcal{P}'$ are horizontal, denoted as $(h_{i,j})_{i,j\in[2k]}$; with horizontal minimality ensured by line segments on the left that intersect the bottom boundary; and vertical minimality ensured by line segments on the bottom (see Figure 5).

The construction of the transversal pair T_k is now similar to Lemma 20. Pick every other segment of $\mathcal{P}'$ for the central vertices, i.e. define $b_{i,j} = h_{2i-1,2j-1}$. Now consider one of the topmost segments $h_{2i,2k}$, and let a_i be the vertical segment that forbids moving $h_{2i,2k}$ left, which is assumed to intersect the bottom boundary of the square. Then a_i and $b_{i',j'}$ intersect if and only if $i \geq i'$. Similarly, we pick c_j to be the vertical segment blocking $h_{2k,2j}$ from moving down. Up to reversing some indices, this yields a transversal pair of order k. ◀

Our main theorem for APUS graphs follows from Lemmas 22, 23, and 25, along the same lines as Theorem 21, with the additional step of splitting into unit squares (Lemma 22). Since computing APUS representation of graphs is NP-hard [16], some subtleties arise regarding the complexity of FO model checking: we assume graphs to be given by their APUS representation, and use that it is easy to compute an APUS representation of transversal pairs as in Figure 5.

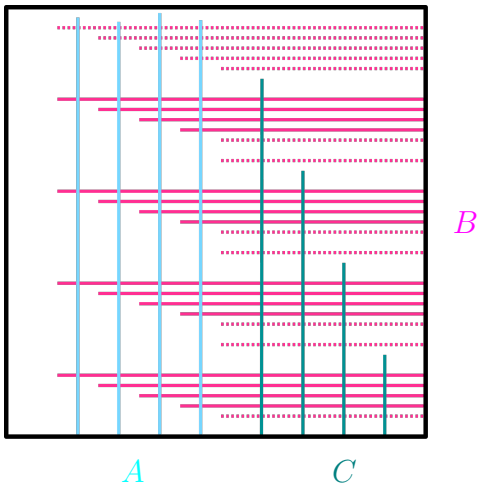


Figure 5 Transversal pair T_4 as APUS graph. The magenta lines are part of an 8-grid. Some of them (solid lines) are picked as B. The others (dotted) are used to define A, C: the cyan (resp. teal) lines forming A (resp. C) ensure horizontal (resp. vertical) minimality of said dotted lines.

▶ **Theorem 26 (♣).** *Let $\mathcal{C}$ be a subclass of non-degenerate APUS graphs. Then $\mathcal{C}$ has bounded twin-width if and only if it is monadically dependent. Further, when $\mathcal{C}$ is hereditary, and assuming* FPT $\neq$ AW[$*$]*, these are also equivalent to FO model checking in $\mathcal{C}$ being FPT, when graphs in $\mathcal{C}$ are given by some APUS representation.*

5 Obstructions to delineation

In this section, we construct graphs with unbounded twin-width, but for which FO model checking is FPT. We use them to show that delineation fails for 1.5D-terrain visibility graphs, and also when relaxing any of the hypotheses of Theorem 26.

Precisely, we construct a class $\mathcal{H}$ of graphs combining half-graphs and paths of slightly more than constant length, and prove that it has unbounded twin-width, but bounded *merge-width* [10]. The latter implies FPT model checking and monadic dependence. The class $\mathcal{H}$ differs significantly from the class of all subcubic graphs used as non-delineation example in [1]: any monadically stable (i.e. that does not transduce all finite linear orders) subclass of $\mathcal{H}$ has bounded twin-width. This is a concrete counter-example to [1, Conjecture 6.6]. Another non-explicit counter-example to this conjecture was given in [6, Corollary 20.10].

Let $\mathfrak{S}_m$ denote the set of permutations of $[m]$. Given $\sigma \in \mathfrak{S}_m$ a length $\ell > 0$, define H_σ^ℓ as follows (see Figure 6). Create vertices a_i, b_i, c_i, d_i for $i \in [m]$, with edges $a_i b_j$ and $c_i d_j$ for all $i \leq j$, thus forming two disjoint half-graphs. Then, connect b_i to $c_{\sigma(i)}$ by a path of length ℓ. For a function $f : \mathbb{N} \to \mathbb{N}$, we define the class of graphs $\mathcal{H}_f := \{H_\sigma^{f(m)} : m \in \mathbb{N},\ \sigma \in \mathfrak{S}_m\}$.

When f grows slowly, we show that $\mathcal{H}_f$ has unbounded twin-width by transducing $f(m)$-subdivisions of cliques of size m, known to have unbounded twin-width [2, Theorem 6.2].

▶ **Fact 27 (♣).** *For $f(m) = o(\log m)$ a positive function, $\mathcal{H}_f$ has unbounded twin-width.*

On the other hand, when f tends to infinity, the class $\mathcal{H}_f$ has bounded merge-width (see full version for the definition and proof). More generally, we prove the following.

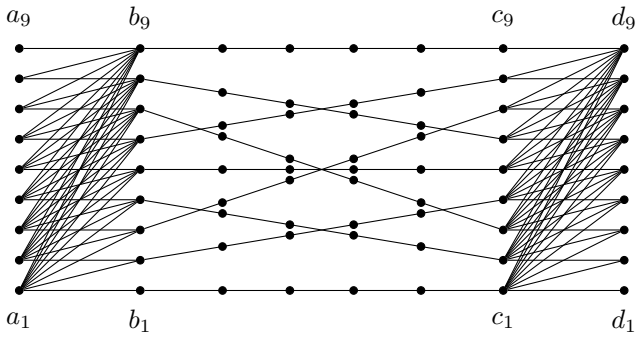


Figure 6 The graph H^5_σ for the permutation $\sigma = 147258369$.

▶ **Lemma 28 (♣).** *Let $\mathcal{C}$ be a class with bounded merge-width, and $f : \mathbb{N} \to \mathbb{N}$ tending to infinity. Call $\mathcal{C}'$ the class of graphs constructed as follows: take $G \in \mathcal{C}$, pick any number of pairs of vertices $(x_1, y_1), \dots, (x_t, y_t)$, and for each add a new path of length more than $f(|V(G)|)$ from x_i to y_i, with fresh internal vertices. Then $\mathcal{C}'$ has bounded merge-width.*

▶ **Fact 29.** *For any function f that tends to infinity, the class $\mathcal{H}_f$ has bounded merge-width.*

Proof. Half-graphs (and disjoint unions thereof) have bounded clique-width, which implies bounded merge-width by [10, Theorem 7.1]. Then, $\mathcal{H}_f$ is obtained from half-graphs by the process described in Lemma 28, hence has bounded merge-width. ◀

Now pick for instance $f(m) \coloneqq \log \log m$, and consider $\mathcal{H} \coloneqq \mathcal{H}_f$.

▶ **Corollary 30.** *The class $\mathcal{H}_f$ has unbounded twin-width, but it is monadically dependent, and FO model checking in $\mathcal{H}_f$ is FPT.*

Proof. The class $\mathcal{H}$ has unbounded twin-width by Fact 27, and bounded merge-width by Fact 29. The latter implies that $\mathcal{H}$ is monadically dependent [10, Theorem 1.12], and has FPT model checking [10, Theorem 1.11]. ◀

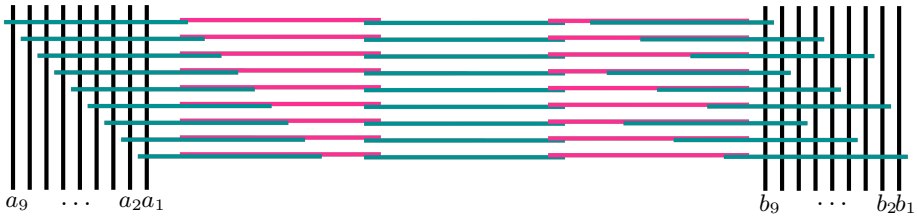


Figure 7 A degenerate APUS whose intersection graph is H^5_σ for $\sigma = 147258369$.

The class $\mathcal{H}_f$ (for any function f with $f(m) \geq 3$) can be constructed as *degenerate* APUS, see Figure 7. Thus the "non-degenerate" assumption in Theorem 26 is necessary. Variants of the construction show that the other hypotheses are necessary too:

▶ **Theorem 31 (♣).** *There are hereditary classes of intersection graphs that have unbounded twin-width but allow FPT FO model checking for the following objects:*

1. *degenerate axis-parallel unit segments,*
2. *axis-parallel segments in general position where segment lengths take only two values,*

3. *axis-parallel segments in general position with lengths in* $[1, 1+\epsilon]$ *for any* $\epsilon > 0$,
4. *unit segments in general position, even when segment slopes are restricted to "vertical", "horizontal plus* ϵ*", and "horizontal minus* ϵ*".*

Case 2 was already known from [1, Figure 8]; we give a different construction based on Figure 7. Finally, a variant of this construction can be used for terrain visibility graphs:

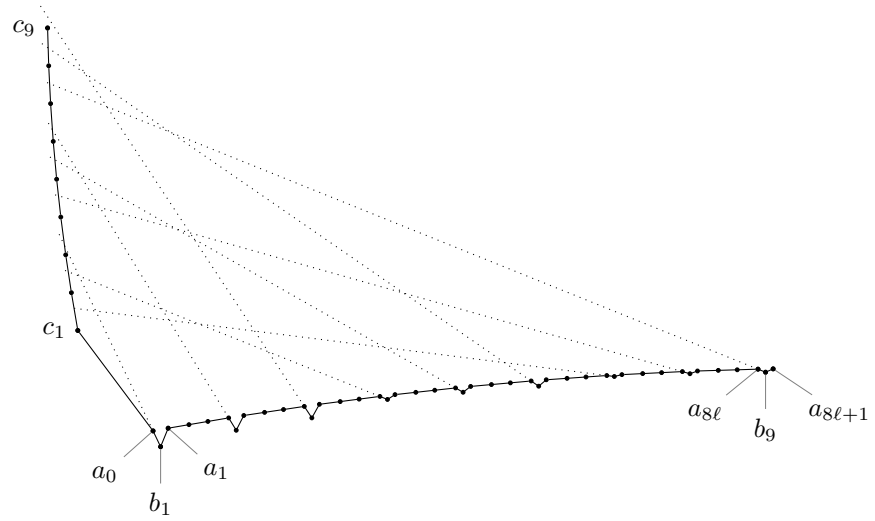


Figure 8 Construction of non-delineated terrain visibility graphs. Dotted lines represent the "horizon" for each b_i, which can be controlled by moving b_i up or down. This allows the half-graph between the b_is and c_js to represent any permutation of the b_is, here 369258147.

▶ **Theorem 2 (♣).** *There is a class* $\mathcal{C}$ *of 1.5D-terrain visibility graphs, whose hereditary closure allows FPT model checking, but which has unbounded twin-width.*

The proof uses one-sided variants of the graphs H^{ℓ}_{σ}, consisting of a long path $a_1, \ldots, a_{\ell n}$, vertices $b_1, \ldots, b_m$ spaced evenly along this path, and a half-graph between $b_1, \ldots, b_m$ and some $c_1, \ldots, c_m$, such that the ordering of $b_1, \ldots, b_m$ from the half-graph can be permuted arbitrarily relative to the path ordering, see Figure 8.

References

1 Édouard Bonnet, Dibyayan Chakraborty, Eun Jung Kim, Noleen Köhler, Raul Lopes, and Stéphan Thomassé. Twin-width VIII: Delineation and win-wins. In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation (IPEC 2022)*, volume 249 of *LIPIcs*, pages 9:1–9:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl. doi:10.4230/LIPIcs.IPEC.2022.9.

2 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width II: small classes. *Combinatorial Theory*, 2(2), 2022. doi:10.5070/C62257876.

3 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Toruńczyk. Twin-width IV: ordered graphs and matrices. *J. ACM*, 71(3), June 2024. doi:10.1145/3651151.

4 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: Tractable FO model checking. *J. ACM*, 69(1):Art. 3, 46, 2022. doi:10.1145/3486655.

5 Flavia Bonomo-Braberman, Nick Brettell, Andrea Munaro, and Daniël Paulusma. Non-crossing h-Graphs: A generalization of proper interval graphs admitting FPT algorithms. In *Graph-theoretic concepts in computer science*, volume 16124 of *Lecture Notes in Comput. Sci.*, pages 121–134. Springer, Cham, [2026] ©2026. doi:10.1007/978-3-032-11835-6_9.

6 Samuel Braunfeld, Jaroslav Nešetřil, Patrice Ossona de Mendez, and Sebastian Siebertz. On first-order transductions of classes of graphs. *Logical Methods in Computer Science*, Volume 21, Issue 2, June 2025. doi:10.46298/lmcs-21(2:26)2025.

7 Derek G. Corneil, Hiryoung Kim, Sridhar Natarajan, Stephan Olariu, and Alan P. Sprague. Simple linear time recognition of unit interval graphs. *Inform. Process. Lett.*, 55(2):99–104, 1995. doi:10.1016/0020-0190(95)00046-F.

8 Rodney G Downey, Michael R Fellows, and Udayan Taylor. The parameterized complexity of relational database queries and an improved characterization of W[1]. *DMTCS*, 96:194–213, 1996.

9 Jan Dreier, Nikolas Mählmann, and Szymon Toruńczyk. Flip-breakability: A combinatorial dichotomy for monadically dependent graph classes. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, pages 1550–1560, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3618260.3649739.

10 Jan Dreier and Szymon Toruńczyk. Merge-width and first-order model checking. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, STOC '25, pages 1944–1955, New York, NY, USA, 2025. Association for Computing Machinery. doi:10.1145/3717823.3718259.

11 Jakub Gajarský, Petr Hliněný, Jan Obdržálek, Daniel Lokshtanov, and M. S. Ramanujan. A new perspective on FO model checking of dense graph classes. *ACM Trans. Comput. Logic*, 21(4), 2020. doi:10.1145/3383206.

12 Colin Geniet and Stéphan Thomassé. First order logic and twin-width in tournaments and dense oriented graphs. *European Journal of Combinatorics*, 132:104247, 2026. doi:10.1016/j.ejc.2025.104247.

13 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *J. ACM*, 64(3):17:1–17:32, 2017. doi:10.1145/3051095.

14 Sylvain Guillemot and Daniel Marx. Finding small patterns in permutations in linear time. In *Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 82–101, 2014. doi:10.1137/1.9781611973402.7.

15 Petr Hliněný and Filip Pokrývka. Twin-width and limits of tractability of FO model checking on geometric graphs, 2022. arXiv:2204.13742.

16 Irina Mustaţă and Martin Pergel. Unit grid intersection graphs: Recognition and properties, 2013. arXiv:1306.1855.

17 Detlef Seese. Linear time computable problems and first-order descriptions. *Mathematical Structures in Computer Science*, 6(6):505–526, 1996. doi:10.1017/S0960129500070079.

18 Alan Tucker. An efficient test for circular-arc graphs. *SIAM Journal on Computing*, 9(1):1–24, 1980. doi:10.1137/0209001.

Online Packing of Orthogonal Polygons

Tim Gerlach
Universität Hamburg, Germany

Benjamin Hennies
Technische Universität Braunschweig, Germany

Linda Kleist
Universität Hamburg, Germany

Abstract

While rectangular and box-shaped objects dominate the classic discourse of theoretic investigations, a fascinating frontier lies in packing more complex shapes. Given recent insights that convex polygons do not allow for constant competitive online algorithms for diverse variants under translation, we study orthogonal polygons, in particular of small complexity. For translational packings of orthogonal 6-gons, we show that the competitive ratio of any online algorithm that aims to pack the items into a minimal number of unit bins is in $\Omega(n/\log n)$, where n denotes the number of objects. In contrast, we show that constant competitive algorithms exist when the orthogonal 6-gons are symmetric or small. For (orthogonally convex) orthogonal 8-gons, we show that the trivial n-competitive algorithm, which places each item in its own bin, is best-possible, i.e., every online algorithm has an asymptotic competitive ratio of at least n. This implies that for general orthogonal polygons, the trivial algorithm is best possible.

Interestingly, for packing degenerate orthogonal polygons (with thickness 0), called skeletons, the change in complexity is even more drastic. While constant competitive algorithms for 6-skeletons exist, no online algorithm for 8-skeletons achieves a competitive ratio better than n.

For other packing variants of orthogonal 6-gons under translation, our insights imply the following consequences. The asymptotic competitive ratio of any online algorithm is in $\Omega(n/\log n)$ for strip packing, and there exist online algorithms with competitive ratios in $O(1)$ for perimeter packing, or in $O(\sqrt{n})$ for minimizing the area of the bounding box. Moreover, the critical packing density is positive (if every object individually fits into the interior of a unit bin).

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Packing, orthogonal polygon, algorithm, offline, online, competitive ratio, bin packing, strip packing, perimeter packing, critical density, 6-gon, 8-gon, L-shape, Z-shape, skeleton

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.52

Related Version *Full Version*: `https://arxiv.org/abs/2603.22098` [30]

Acknowledgements We thank the anonymous reviewers for their valuable comments and suggestions, which helped to improve the quality of this work. In particular, we thank an anonymous reviewer for the suggestion to use NextFit in the proof of Theorem 6(i) and improving the analysis to obtain a competitive ratio of 2.

1 Introduction

Packing problems not only present us with constant challenges in everyday life, but also find applications in manufacturing industries, logistics, and scheduling. While rectangular and box-shaped objects dominate the classic discourse of theoretic investigations, a fascinating frontier lies in the packing of more complex objects, e.g., convex polygons. The problem of packing convex polygons is particularly interesting when restricting the allowed motions to translations, as allowing for rotations reduces the problem to mere rectangle packing. Across the numerous variants – be it strip packing, bin packing, or perimeter packing – the overall

© Tim Gerlach, Benjamin Hennies, and Linda Kleist;
licensed under Creative Commons License CC-BY 4.0

picture is that constant factor approximations exist for translational packings of convex polygons [7, 8, 1, 41]. However, the landscape changes when we step into the online realm where items arrive sequentially one after the other and decisions have to be made irrevocably without knowing the future items: In this setting, no online algorithm for translational packings of convex polygons has a constant competitive ratio, as recently shown by Aamand, Abrahamsen, Beretta, and Kleist [1]. This is in stark contrast to the case of rectangles, for which constant competitive algorithms exist for various variants, cf. Section 1.1.

In this work, we study the online packing problem of orthogonal polygons which are orthogonally convex, combining relaxed features of both orthogonality and convexity. A polygon is *orthogonally convex* if the intersection with any axis-parallel line has at most one connected component. The simplest yet unresolved setting involves orthogonal 6-gons, which include *L-shapes*. For examples, consider Figure 1.

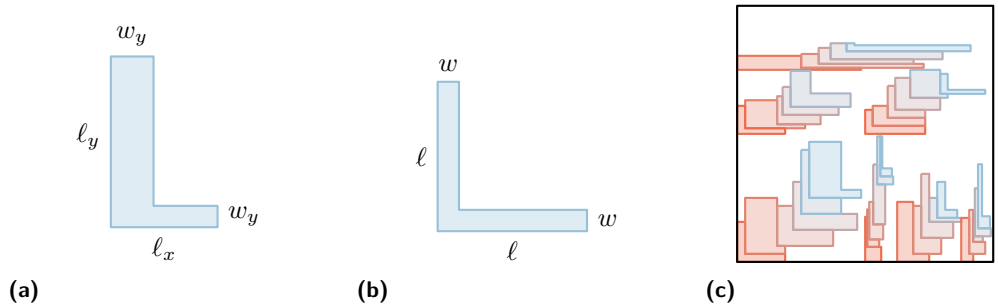


Figure 1 (a) An L-shape, (b) a symmetric L-shape, and (c) a bin packing of L-shapes.

The diverse applications motivate a multitude of interesting packing problems. Among them are bin packing, strip packing, perimeter packing and area packing for various types of objects and allowed motions; for definitions see Section 2. Many variants of these packing problems are known to be computationally difficult. While NP-hardness has usually been known for a long time [5, 28, 35, 43], more recently, some variants were proved to even be $\exists\mathbb{R}$-complete by Abrahamsen, Miltzow, and Seiferth [4]. Adding to this difficulty, we might not have complete information about the objects in advance, e.g., they may appear over time. This is captured by the so called *online setting* where the objects are presented one after another and the next object is only revealed when the previous one has been placed.

In *competitive analysis*, the performance of an online algorithm is measured against the optimal offline solution. While the *absolute competitive ratio* gives the worst-case performance over all possible inputs, the *asymptotic competitive ratio* describes the behavior when the input size goes to infinity. For precise definitions, we refer to Section 2. In the following we present our contributions, see also Tables 1 and 2. Usually, we present lower bounds on the asymptotic competitive ratio and upper bounds on the absolute competitive ratio which imply the statements for both ratios.

Bin Packing of 6-gons. For bin packing of orthogonal polygons, it is natural to consider the variants where the objects can only be translated, and where the objects may be rotated by multiples of 90 degrees. The following proposition shows that even in the latter variant we may restrict our analysis to the translational case. In the problem Ortho-BinPack, we receive orthogonal polygons $P_1, ..., P_n$ which have to be placed into a unit square bin by translation. The goal is to minimize the number of used bins. Ortho-BinPack$_{\text{rot}}$ denotes the variant where the polygons might be rotated by multiples of 90 degrees.

▶ **Proposition 1.** *The best (absolute/asymptotic) competitive ratios for* Ortho-BinPack *and* Ortho-BinPack$_{rot}$ *are within constant factors of each other. The analogous statements hold for the best approximation ratios of the offline variants.*

Proposition 1 allows us to focus on translations only. We start with a lower bound on the competitive ratio of any online algorithm for L-shapes. To this end, let L-BinPack denote the variant of Ortho-BinPack where each polygon is an L-shape.

▶ **Theorem 2.** *The asymptotic competitive ratio of any online algorithm for L-BinPack is in* $\Omega(n/\log n)$.

We obtain this lower bound by reducing from an online sorting problem where natural numbers have to placed in arrays such that the numbers in each array are increasing. Note that the trivial algorithm, which packs each object into its own bin, is n-competitive.

In contrast to Theorem 2, there exist constant competitive algorithms for some interesting subclasses of L-shapes, namely when the L-shapes are either all symmetric or all small. An L-shape is described by four parameters $\ell_x, \ell_y, w_x, w_y \in [0, 1]$. It is *small* if $\ell_x, \ell_y \leq 1/2$, *large* if $\ell_x, \ell_y \geq 1/2$, and *symmetric* if $\ell_x = \ell_y$ and $w_x = w_y$. For illustrations see Figure 1.

▶ **Theorem 3.** *There is an online algorithm for L-BinPack with a constant asymptotic competitive ratio when all L-shapes are symmetric or small. In particular, there exist algorithms that use at most*

(a) $33 \cdot OPT + 2$ *bins when all L-shapes are large and symmetric.*

(b) $8 \cdot OPT + 7$ *when all L-shapes are small.*

(c) $41 \cdot OPT + 9$ *when all L-shapes are symmetric.*

Theorem 3 directly yields algorithms with a constant absolute competitive ratio of 35, 15 and 50, respectively, as in bin packing, we have $\mathrm{OPT} \geq 1$ for any nonempty set of items.

The results for symmetric L-shapes are based on identifying their connection to scheduling problems. Competitive algorithms for two special cases of large and symmetric L-shapes follow directly from previous work on scheduling problems. Precisely, an e-competitive algorithm for objects of equal widths [22] and a 2-competitive algorithm for the case of equal lengths [24] can be deduced, see Subsection 1.1 for explanations. A connection to scheduling problems is also used for the 33-competitive algorithm which builds on a constant competitive algorithm by Devanur et al. [22] for a special machine minimization problem in combination with an online coloring algorithm for intervals graphs by Kierstead and Trotter [39]. For details on the connection to scheduling problems, we refer to Subsection 1.1.

Table 1 Bounds on the best asymptotic competitive ratios of bin packing orthogonal polygons.

Objects		Lower Bound		Upper Bound	
▯	rectangles	1.91	[23]	2.56	[31]
⌊	symmetric L-shapes	1.91	[23]	41	[Thm 3c]
⌊	orthogonal 6-gons	$n/\log n$	[Thm 2]	n	
⌊	orthogonal 8-gons	n	[Thm 5]	n	
⌊	L-skeletons	1.54	[12]	2	[Thm 6]
⌐	Z-skeletons	n	[Thm 6]	n	

Other packing variants. Our insights on online bin packing of L-shapes have consequences for other packing variants. In the problem L-PeriPack, we receive L-shapes $L_1, ..., L_n$, and the goal is to place them in the plane such that the perimeter of the bounding box is minimized. In L-AreaPack, the goal is to minimize the area of the bounding box. For a unit square bin and L-shapes with arm lengths ℓ_x, ℓ_y bounded by some constant $t < 1$, the *online critical density* is the largest value of A such that there exists an online algorithm which packs any sequence with total area at most A into the bin.

▶ **Corollary 4.** *The following statements hold for L-shapes with arm lengths in* $[0, 1]$*:*

- **(i)** *The asymptotic competitive ratio of any online algorithm for L-StripPack is in* $\Omega(n/\log n)$.
- **(ii)** *There is an online algorithm for L-PeriPack with absolute competitive ratio in* $O(1)$.
- **(iii)** *There is an online algorithm for L-AreaPack with asymptotic competitive ratio in* $O(\sqrt{n})$.
- **(iv)** *The online critical packing density is positive if the arm lengths of each L-shape are bounded by* $t < 1$. *For* $t = 1$, *the online critical packing density is* 0.

The $O(\sqrt{n})$-competitive algorithm for L-AreaPack is asymptotically best-possible, as a lower bound of $\Omega(\sqrt{n})$ already for rectangles has been shown by Abrahamsen and Beretta [3].

Table 2 Overview on the known upper and lower bounds on the best competitive ratios for the various packing problems.

Variant	▯ rectangles	△ convex polygons	⌙ L-shapes	⌐ ortho polygons
Bin	$\Theta(1)$ [34]	$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$ [1]	$\Omega\left(\frac{n}{\log n}\right)$ [Thm 2]	n
Strip	$\Theta(1)$ [52]	$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right), O(n^{0.59})$ [1]	$\Omega\left(\frac{n}{\log n}\right)$ [Thm 4i]	$\Omega\left(\frac{n}{\log n}\right)$ [Thm 4i]
Perimeter	$\Theta(1)$ [3]	$\Omega\left(\sqrt[4]{\frac{\log n}{\log\log n}}\right)$ [1]	$\Theta(1)$ [Thm 4ii]	
Area	$\Theta(\sqrt{n})$ [3]	$\Omega(\sqrt{n})$ [3]	$\Theta(\sqrt{n})$ [3,Thm 4iii]	$\Omega(\sqrt{n})$ [3]

Orthogonal polygons of higher complexity. For bin packing of orthogonal 8-gons, we show that the trivial algorithm, that places each shape into its own bin, achieves the best possible competitive ratio. Hence, the trivial algorithm is also best possible for Ortho-BinPack.

▶ **Theorem 5.** *There is no online algorithm for Ortho-BinPack that achieves an asymptotic competitive ratio better than* n, *even when restricting to orthogonally convex 8-gons.*

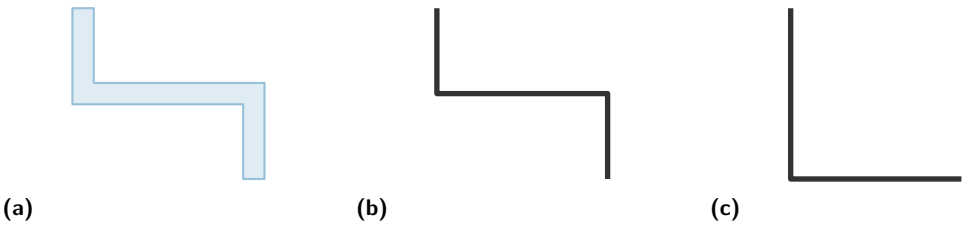


Figure 2 (a) A Z-shape (b) A Z-skeleton (c) An L-skeleton.

We obtain Theorem 5 by considering Z-shapes of equal thickness; Figure 2a depicts an example. Interestingly, we obtain this result via first considering packings of skeletons, which can be viewed as orthogonal polygons of widths 0, see Figures 2b and 2c. In the problem L-Skel-BinPack (Z-Skel-BinPack), L-skeletons (Z-skeletons) shall be packed into a minimum number of bins. In this context, we consider a packing of skeletons as *valid* if any two objects are disjoint, with the exception that either of the two endpoints of one skeleton may touch another; a point of a Z-skeleton is an endpoint if it is a unique extreme point, i.e., topmost, bottommost, leftmost, or rightmost. From L- to Z-skeletons, there is a quite drastic jump in complexity.

▶ **Theorem 6.** *For online bin packing of orthogonal skeletons the following holds.*

(i) *There exists an online algorithm for L-Skel-BinPack with a constant absolute competitive ratio.*

(ii) *No online algorithm for Z-Skel-BinPack has an asymptotic competitive ratio better than n.*

Organization. The remainder of the paper is organized as follows. In Section 1.1, we discuss related work. In Section 2, we give a concise dictionary on crucial notions. In Section 3, we prove Theorem 2, i.e., the lower bound on the competitive ratio for L-BinPack and use our insights to show a lower bound for L-StripPack in Section 4. In Section 5, we consider orthogonal polygons of higher complexity and sketch the proofs of Theorems 5 and 6(ii). Complete proofs of all results can be found in the full version of this paper [30].

1.1 Related work

As the literature on online packing problems is extensive, we refer to the surveys of Christensen, Khan, Pokutta, and Tetali [19], Epstein and van Stee [26], van Stee [49, 50], and Csirik and Woeginger [21] for an overview. Below we discuss the most important results related to the packing problems studied in this paper.

Rectangles. Rectangle packings has been studied for almost 50 years. A straightforward reduction from partition shows that the offline strip packing problem cannot be approximated with an absolute factor better than $3/2$ unless $\mathsf{P} = \mathsf{NP}$, and the best known approximation ratio is $5/3 + \varepsilon$ by Harren, Jansen, Prädel, and Van Stee [35]. For the online variant, first fit shelf algorithms as proposed by Baker and Schwarz [10], are widely studied. The best known competitive ratio of $7/2 + \sqrt{10} \approx 6.66$ is achieved by Ye, Han, and Zhang [52]. Restricting the attention to large instances, the best asymptotic competitive ratio is lower bounded by 1.54 by van Vliet [51] and upper bounded by 1.59 by Han, Iwama, Ye, and Zhang [32].

For *bin packing* of rectangles under translation, there exist approximation algorithms with approximation ratio 2 [34] and asymptotic approximation guarantee 1.4 by Bansal and Khan [14]. For online bin packing of rectangles, the upper bound on the asymptotic competitive ratio for online translational bin packing axis-parallel rectangular pieces into unit square bins has been decreased in a series of papers from 3.25 [20] to about 2.55 by Han, Chin, Ting, Zhang and Zhang [31] and the lower bound has been increased from 1.6 [29] to 1.91 by Epstein [23]. For the special case of squares, the lower bound is approximately 1.75 due to Balogh, Békési, Dósa, Epstein, and Levin [11], while the upper bound is above 2.08 by Epstein and Mualem [25].

For the problem of area minimization, Abrahamsen and Beretta [3] present $O(n)$-competitive algorithms and show a lower bound of $\Omega(n)$ for both versions of rotation allowed and translations only. For perimeter minimization, they gave a 3.98-competitive algorithm for both, translation and rotation, as well as a lower bound on the competitive ratio of $4/3$ for the case of translation and of $5/4$ for the case of rotation.

Convex Polygons. Constant-factor approximations for packing convex polygons exist for many variants [7, 8, 1, 41]; an exception is bin packing where constant approximation factors are only known if the diameter of the items is bounded by a constant. In contrast to these constant guarantees, for online translational packings of convex polygons, Aamand, Abrahamsen, Beretta and Kleist [1] show that strip packing, bin packing, and perimeter packing do not allow for constant competitive online algorithms, even if all pieces have a diameter bounded by an arbitrarily small constant. To this end, they reduce to an online sorting problem which sparked follow-up investigations [2, 9, 47]. Concerning upper bounds, they give an online algorithm with competitive ratio $O(n^{0.59})$ for strip packing. The knapsack problem for convex polygons under rigid motions is studied by Merino and Wiese [45].

Orthogonal Polygons and Scheduling. We identified interesting connections between packings of special L-shapes and scheduling problems such as open end bin packing and machine minimization. These yield algorithms for large and symmetric L-shapes in the case where all L-shapes have equal lengths or equal widths; for an overview see Table 3.

The case of equal lengths is related to open end bin packing. In a variant of classical 1D bin packing known as *ordered open end bin packing*, items of sizes in $(0, 1]$ are presented one by one, and must be assigned to bins in this order. An item can be assigned to any bin for which the current total size is strictly below 1, i.e., the bin can be overloaded by its last packed item. Balogh, Epstein, and Levin [13] present a 2-competitive online algorithm and Epstein [24] designs a 1.5 approximation. As packings of large, symmetric L-shapes of equal lengths can be reduced to open end bin packing, we obtain offline and online algorithms with constant factors. Specifically, for packing L-shapes of length ℓ, we consider open end bins of capacity $1 - \ell$. An L-shape L_i with arm length ℓ and width w_i is associated with a real $r_i := w_i$. It is easy to see that an L_i can be packed into a bin containing $L_1, \ldots, L_k$ (bottom-leftmost) if $\sum_{j=1}^{k} w_j + \ell \leq 1$. Similarly, r_i can be packed into an open bin containing $r_1, \ldots, r_k$ if (and only if) $\sum_{j=1}^{k} r_j \leq 1 - \ell$. As these conditions are equivalent, solutions of open end bin packing translate to (bottom-leftmost) packings of large, symmetric L-shapes of equal lengths. This implies a 2-competitive online algorithm and a 1.5-approximation in this case.

For bin packing of large and symmetric L-shapes with different lengths, we identified a close connection to the *machine minimization* problem: Given a set of jobs, each of which has a release time r, a processing time p and deadline d, the task is to schedule these jobs on a minimum number of machines such that each machine processes at most one job at a time and each job starts not before its release time, and finishes not after its deadline. For large and symmetric L-shapes, we show that the two parameters, length ℓ and width w, translate to a job with release time 0, processing time w, and deadline $1 - \ell$. Therefore, we present relevant work in the context of machine minimization. Yu and Zhang [53] present approximation algorithms for the special cases of equal deadlines and equal processing times. In particular, their 2-approximation algorithm for the special case of equal deadlines translates to a 2-approximation for offline bin packing of large, symmetric L-shapes. Kao, Chen, Rutter, and Wagner [38, Lemmas 1 and 2] present a polynomial algorithm that computes the offline

optimal solution for jobs of unit processing times. This translates to a polyomial time algorithm for offline packings of instances where all L-shapes are large, symmetric, and have equal widths. Devanur, Makarychev, Panigrahi, and Yaroslavtsev [22, Section 2,3] present an e-competitive online algorithm for unit processing times; they also show that the competitive ratio of e is optimal. This result is also implied by Bansal, Kimbrel, and Pruhs [15, Lemmas 4.7 and 4.8]. The insights can be translated to an e-competitive algorithm for packing large, symmetric L-shapes of equal widths. Moreover, Devanur et al. [22] present a 16-competitive online algorithm for the case with equal deadlines; their techniques are the basis of our 33-competitive algorithm for packing large symmetric L-shapes, namely Theorem 3(a).

There are many more results on the general case of machine minimization. However, due to a lower bound of n by Saha [48], most subsequent work allows preemption of the jobs [18, 17, 36], a setting that does not carry over to packings of L-shapes.

Table 3 Upper bounds on the best asymptotic approximation/competitive ratios for L-BinPack with restricted L-shapes.

Type	Offline		Online	
large, symmetric	2	[53]	33	[Thm 3a]
+ equal length	1.5	[24]	2	[13]
+ equal width	1	[38]	e	[22]
small	8	[Thm 3b]	8	[Thm 3b]

Critical densities. For a given container of volume C and a class of objects in $\mathbb{R}^d$, the *critical density* is the largest value of V/C such that any sequence of objects of the class with a total volume of at most V can always be packed in the container. For convex bodies of bounded diameter in $\mathbb{R}^3$, Auerbach, Banach, Mazur and Ulam [44] stated (without proof) that the critical density is positive, i.e., there exists a function f such that any sequence of convex bodies in $\mathbb{R}^3$, each of diameter $\leq \delta$ and total volume of at most V, can be packed into a cube with side length $s = f(\delta, V)$ when rotations are allowed. The first proof, even for arbitrary dimension, is given by Kosiński [40]. When rotations are not allowed, Alt, Cheong, Park, and Scharf [6] showed that the critical density of packing convex bodies of bounded diameter into a cube is 0; in particular for packing unit disks in $\mathbb{R}^3$. However, for cubes, Moon and Moser [46] showed that any sequence of cubes in $\mathbb{R}^d$ of total volume $1/2^{d-1}$ can be packed into the unit cube. This bound is the best possible, because for any $\varepsilon > 0$, two cubes with side lengths $1/2 + \varepsilon$ cannot be packed in the unit cube.

The study of critical densities likewise makes sense when the pieces appear in an online fashion. A lower bound on the critical density of online packing squares into the unit square has been improved in a sequence of papers [37, 16, 33, 27] from 5/16 [37] to 2/5 [16]. Interestingly, Januszewski and Lassak [37] proved that in dimension $d \geq 5$, the critical density of online packing cubes into the unit cube is $1/2^{d-1}$, just as in the offline case. Lassak and Zhang [42] showed that for some constant $\delta(d) > 0$, any sequence of axis-parallel boxes of diameter and total area at most $\delta(d)$ can be packed online in the d-dimensional unit hypercube using translations. This implies that the critical packing density for convex bodies is positive for any dimension $d \geq 1$ when rotations are allowed as each body can be rotated so that it has a constant (depending on the dimension) density in its axis-parallel bounding box. In contrast, for translational and online packing convex polygons, Aamand, Abrahamsen, Beretta and Kleist [1] show that the critical packing density is 0.

2 Notions: packing variants, competitive analysis, and objects

We briefly define the packing variants, review the common terminology for competitive analysis, and introduce the considered objects.

Packing variants. Depending on the context, there are numerous packing variants specified by the container and objective. The variants discussed in this paper are as follows. Consider a set (or sequence) of items. Given an unbounded supply of identical unit bins, the goal of *bin packing* is to pack the items into as few bins as possible. Given a horizontal strip of height 1 which is bounded to the left but infinite to the right, the aim of *strip packing* is to place the items in the strip such that the maximum x-coordinate of an occupied point is minimized. In *perimeter packing* or *area packing*, the goal is to find a placement of the items in the plane such that the perimeter or the area of the bounding box is minimized, respectively. All of these variants are interesting as *offline* and *online* problems. In this work we focus on online problems and therefore review competetitive analysis.

Competitive analysis. In an online problem, the input is a sequence $\sigma_1, \ldots, \sigma_n$ of objects, and we need to process object σ_i (in a problem-specific manner) before the next object σ_{i+1} is revealed. In this work, the objects are either natural numbers, orthogonal polygons, or their skeletons. We briefly revisit the standard terminology of competitive analysis for an online algorithm $\mathcal{A}$ of a minimization problem. For an instance I, $\mathrm{OPT}(I)$ denotes the cost of the offline optimum solution and $\mathcal{A}(I)$ denotes the cost of the solution computed by $\mathcal{A}$ for input I. Let $f\colon \mathbb{N} \to \mathbb{R}$ be a function of the size $|I|$ of the instance, in our context typically the number of objects. We say that $\mathcal{A}$ has *(absolute) competitive ratio* $f(|I|)$ if, for all instances I, it holds that $\mathcal{A}(I) \leq f(|I|) \cdot \mathrm{OPT}(I)$. Similarly, $\mathcal{A}$ has *asymptotic competitive ratio* $f(|I|)$ if there exists a constant $\beta > 0$ such that $\mathcal{A}(I) \leq f(|I|) \cdot \mathrm{OPT}(I) + \beta$ for all I.

Objects. We focus on packings of orthogonal polygons that we call *L-shapes* and *Z-shapes*. An *L-shape* is the union of a rectangle of width ℓ_x and height w_y with a rectangle of width w_x and height ℓ_y, with $w_x \leq \ell_x$ and $w_y \leq \ell_y$, such that their lower left corners coincide. Figure 3a presents an example. We describe the placement of an L-shape by specifying the coordinates of its *reference point*, namely its lower left corner.

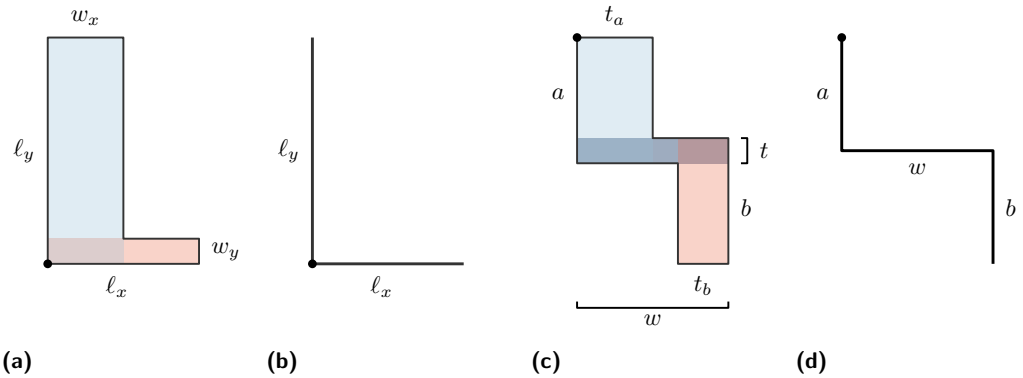


Figure 3 Parameters and the reference point of an (a) L-shape, (b) L-skeleton, (c) Z-shape, and (d) Z-skeleton.

We note that each orthogonal 6-gon is either an L-shape or a rotated L-shape. An L-shape is *small* if $\ell_x, \ell_y \leq 1/2$, *large* if $\ell_x, \ell_y \geq 1/2$, and *symmetric* if $\ell_x = \ell_y$ and $w_x = w_y$.

A *Z-shape* is the union of three rectangles: A base rectangle of height t and width w, and two arm rectangles: One arm (above) of width t_a and height a, whose lower left corner coincides with the lower left corner of the base, and one arm (below) of width t_b and height b such that its upper right corner coincides with the upper right corner of the base, where $t_a, t_b \leq w$, see also Figure 3c. In this paper, we will mostly be concerned with Z-shapes of *equal thickness*, i.e., the case where $t_a = t_b = t$. Note that all L-shapes and Z-shapes are *orthogonally convex*, i.e., the intersection with any axis-parallel line is connected.

L-skeletons and Z-skeletons are the corresponding shapes with zero thickness, i.e., $w_x = w_y = 0$ and $t_a = t_b = t = 0$, respectively. Figures 3b and 3d present examples.

3 Lower bound for bin packing of L-shapes – Proof of Theorem 2

We now prove the lower bound for L-BinPack.

▶ **Theorem 2.** *The asymptotic competitive ratio of any online algorithm for L-BinPack is in* $\Omega(n/\log n)$.

We show this in two steps. We prove that any online algorithm for L-BinPack yields an algorithm for an online sorting problem called BinSorting with the same competitive ratio, and show that every online algorithm for BinSorting has a competitive ratio in $\Omega(n/\log n)$. We start by introducing the problem BinSorting.

3.1 Bin sorting

The online problem BinSorting$[k]$ can be described as a game between two players, Presenter and Algorithm, playing on several arrays with k slots each, in which natural numbers have to be inserted. Within each array, the numbers must be increasing. Presenter presents distinct numbers. When receiving a number, Algorithm has to irrevocably decide in which array and in which slot the number is placed while obeying the rule that the numbers of each array are increasing. The goal of Algorithm is to minimize the number of used arrays, while Presenter wants to maximize it. Figure 4 presents an example of this game.

	2_4	16_1		
4_3	8_2			
		3_5		

Figure 4 An instance of the BinSorting with $k = 5$ for the sequence $16, 8, 4, 2, 3$.

The offline optimum corresponds to the setting where Algorithm knows all numbers in advance and is simple to determine: For n numbers in arrays with k slots, $\lceil n/k \rceil$ arrays are necessary and, as the numbers can be inserted in a sorted manner, also sufficient. Consequently, $\mathrm{OPT} = \lceil n/k \rceil$.

We start by presenting a strategy for Presenter forcing Algorithm to use at most a logarithmic number of slots in each array.

▶ **Lemma 7.** *The asymptotic competitive ratio of any online algorithm for BinSorting$[k]$ is at least* $\lceil n/\lfloor \log(k+1) \rfloor \rceil / \lceil n/k \rceil \in \Omega(k/\log k)$. *For $k = n$, the lower bound is* $n/\lfloor \log(n+1) \rfloor \in \Omega(n/\log n)$.

Proof. We show that PRESENTER has a strategy to present n numbers such that ALGORITHM uses at least $n/\lfloor\log(k+1)\rfloor$ arrays while $\lceil n/k\rceil$ arrays would suffice. The idea is as follows: Consider a partially filled array A. The numbers in A partition it into maximal sections of free slots. If a new number shall be inserted in A, it may be placed into at most one of these sections, since the entries in the array must be increasing. We say that two (possible future) numbers are *similar* with respect to A if they must be placed in the same section. The idea of PRESENTER is to maintain a set of active numbers that are similar for all arrays.

In order to present n natural numbers $a_1, \dots, a_n$, PRESENTER maintains the following invariant: In the beginning of iteration i, it has an *active* set of $2^{n-i+1}-1$ similar numbers. In the very beginning (of iteration 1), the set is $(0, 2^n) := \{1, \dots, 2^n - 1\}$, each array has one section containing all slots, and PRESENTER presents $a_1 = 2^{n-1}$. In iteration $i+1$, PRESENTER determines the next number to be presented based on the action of ALGORITHM in iteration i. ALGORITHM places a_i in a section of some array which results in two smaller sections. If the smaller of the two section lies to the left of a_i, then the active set (l, r) is updated to (l, a_i) and $a_{i+1} = (l + a_i)/2 = a_i - 2^{n-(i+1)}$; otherwise it is updated to (a_i, r) and $a_{i+1} = (a_i + r)/2 = a_i + 2^{n-(i+1)}$. As the new active set is a subset of the previous, ALGORITHM must place numbers presented in the future in (subsets of) the current sections of the active set. By construction, the numbers of the active set remain similar. The active set is non-empty for at least n iterations because it contains $2^{n-i+1} - 1 \geq 1$ numbers if $i \leq n$.

When we restrict our attention to a fixed array with k slots, then the section for the active numbers is (at least) halved each time a number is inserted; this is because we ensure that the active set must be inserted in the smaller section. Therefore, ALGORITHM can place at most $\lfloor \log(k+1) \rfloor$ of the n numbers in each array and thus needs at least $\lceil n/\lfloor\log(k+1)\rfloor\rceil$ bins. This yields a competitive ratio of at least $\lceil n/\lfloor\log(k+1)\rfloor\rceil / \lceil n/k\rceil \in \Omega(k/\log k)$. ◀

We remark that the lower bound of Lemma 7 is tight, that is, there exists an online algorithm BINSORTING$[k]$ that matches this bound.

▶ **Lemma 8.** *There exists an online algorithm for* BINSORTING$[k]$ *with absolute competitive ratio of* $\lceil n/\lfloor\log(k+1)\rfloor\rceil / \lceil n/k\rceil \in O(k/\log k)$.

Proof. For each partially filled array and any number, the valid slots form a consecutive interval. Clearly, in an empty array, all slots are valid. Let $\mathcal{A}$ denote the algorithm that places a new number in the first array with a valid slot, and in particular, in the middle slot of the section. With this choice, the smallest section of an array containing m numbers has length $\geq \lfloor (k+1)/2^m \rfloor - 1$. Therefore each array contains at least $\lfloor\log(k+1)\rfloor$ numbers before a new array is opened. Consequently, $\mathcal{A}$ uses at most $\lceil n/\lfloor\log(k+1)\rfloor\rceil$ arrays while $\mathrm{OPT} = \lceil n/k\rceil$. ◀

3.2 From L-BinPack to BinSorting

We now show how to create instances of L-BINPACK which simulate BINSORTING. The construction is visualized in Figure 5. For fixed $k \in \mathbb{N}$, we consider the family $\mathcal{L}^k = \{L_i\}_{i\in\mathbb{N}}$ of L-shapes with the following parameters:

$$\ell_x(L_i) = 1/2 + 1/2k, \quad w_x(L_i) = 1/2k, \quad \ell_y(L_i) = 1 - 2^{-i-1}, \quad w_y(L_i) = 2^{-i-1}$$

The key idea is that L_i in L-BINPACK mimics the natural number i for BINSORTING. We start with the crucial property that each subset of $\mathcal{L}^k$ packed into a bin is sorted.

▶ **Lemma 9.** *Each set $S \subset \mathcal{L}^k$ packs into one bin if and only if $|S| \leq k$ and the items are placed in descending order of their indices i.*

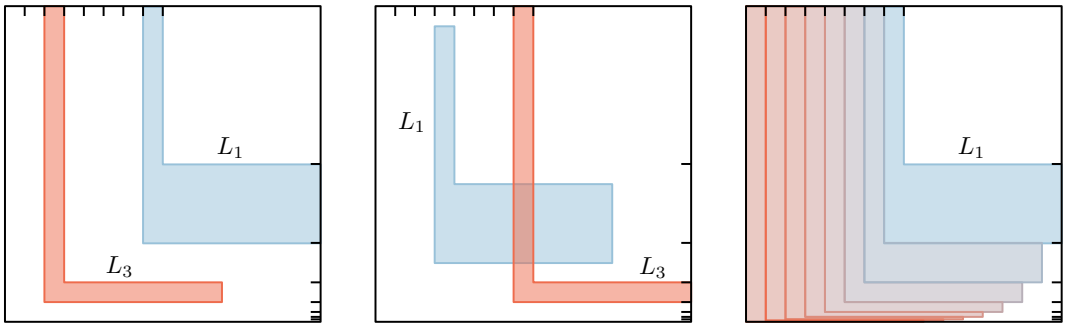


Figure 5 Illustrations for $\mathcal{L}^8$. (a) While L_1 may be placed to the right/above of L_3, (b) L_1 cannot be placed to the left/below of L_3 as $l_x(L_1)+l_x(L_3) > 1$ and $l_y(L_3)+w_y(L_1) = 1-2^{-3-1}+2^{-1-1} > 1$, respectively. (c) A packing of $L_1, \dots, L_k \in \mathcal{L}^k$ for $k = 8$ into one bin.

Proof. By construction up to k items of $\mathcal{L}^k$ fit into one bin in descending order of their indices: With respect to the x-coordinate, exactly k items can be packed next to each other. For the y-coordinate, we note the following. When L_i is placed somewhere in the bin, the free space above is lower bounded by

$$\ell_y(L_i) - w_y(L_i) = (1 - 2^{-i-1}) - 2^{-i-1} = \ell_y(L_{i-1}) \geq \ell_y(L_j) \text{ for } j < i.$$

So when placed in descending order, there is space for k items. Figure 5c depicts an example for $k = 8$. It remains to show that L_i cannot be placed below/left of L_j when $i < j$. It cannot be placed entirely to the left of L_j, because $l_x(L_i) + l_x(L_j) = 2(1/2 + 1/k) > 1$, and it cannot be placed below L_j because $w_y(L_i) + \ell_y(L_j) = 2^{-i-1} + 1 - 2^{-j-1} > 1$. ◀

We now show how an algorithm for L-BinPack yields an algorithm for BinSorting$[k]$.

▶ **Lemma 10.** *An online-algorithm for L-BinPack yields an algorithm for BinSorting$[k]$ with the same (absolute/asymptotic) competitive ratio.*

Proof. Let $\mathcal{A}$ be an α-competitive algorithm for L-BinPack and consider an instance I_S of BinSorting $[k]$ with n numbers. We describe an online algorithm $\mathcal{A}'$ for BinSorting$[k]$ which uses $\mathcal{A}$ as a subroutine. For each presented number i, $\mathcal{A}'$ presents $L_i \in \mathcal{L}^k$ to $\mathcal{A}$. By Lemma 9, at most k L-shapes of $\mathcal{L}^k$ fit into one bin. Let x denote the x-coordinate of the reference point of L_i in the jth bin as assigned by $\mathcal{A}$. Then $\mathcal{A}'$ places number i in the slot $\lfloor x \cdot 2k \rfloor + 1$ of the jth array. As $x \in [0, 1/2 - 1/2k]$, it holds that $\lfloor x \cdot 2k \rfloor + 1 \in [k]$. We shortly argue that this position is valid. Firstly, because all L-shapes have width $1/2k$, the assigned x-values differ by at least $1/2k$ and for any x and x' with $|x - x'| \geq 1/2k$, we have $|\lfloor x \cdot 2k \rfloor - \lfloor x' \cdot 2k \rfloor| \geq 1$. Secondly, as $\mathcal{A}$ guarantees that the numbers of each array are increasing, the same holds for the indices of the L-shapes. Hence, by Lemma 9, the placement is valid. Consequently, we may translate the x-position of L_i in the jth bin to a unique slot in the jth array. The number of used arrays equals the number of used bins. Moreover, the offline optimum is $\lceil n/k \rceil$ in both problems. Hence, $\mathcal{A}'$ is α-competitive. ◀

Together, Lemmas 7 and 10 show the following statement, which proves Theorem 2.

▶ **Corollary 11.** *For each online algorithm $\mathcal{A}$ for L-BinPack and each n, there exists a strategy to present n L-shapes from $\mathcal{L}^n$ (no duplicates) such that they can all be packed into one bin while $\mathcal{A}$ uses at least $n/\log(n+1)$ bins.*

4 Strip packing of L-shapes – Proof of Corollary 4(i)

We now prove Corollary 4(i). To this end, we show how a strip packing algorithm yields a bin packing algorithm.

▶ **Lemma 12.** *An algorithm $\mathcal{A}$ for strip packing a sequence of n L-shapes from $\mathcal{L}^n$ (without duplicates) with asymptotic competitive ratio α yields an algorithm $\mathcal{A}'$ for L-BinPack with asymptotic competitive ratio 2α.*

Proof. Let $\mathcal{A}$ be an asymptotic α-competitive algorithm for L-StripPack and consider an instance of L-BinPack with n objects from $\mathcal{L}^n$. We describe an online algorithm $\mathcal{A}'$ for L-BinPack which uses $\mathcal{A}$ as a subroutine. For each each $L \in \mathcal{L}^n$, $\mathcal{A}'$ presents it to $\mathcal{A}$ and observes its placement in the strip. The idea is to partition the strip into slots of widths $1/2n$; each slot will correspond to a placement in a bin. For an illustration, see Figure 6. We show that objects of n consecutive slots can be packed into one bin. If the reference point of L is placed at (x, y) in the strip, then $\mathcal{A}'$ places it as follows. Let $m, r \in \mathbb{N}$ such that $\lfloor x \cdot 2n \rfloor = mn + r$ and $r < n$. Then $\mathcal{A}'$ places L in the $(m+1)$st bin such that its reference point is at $(x', y') := (r/2n, 1 - \ell_y(L))$.

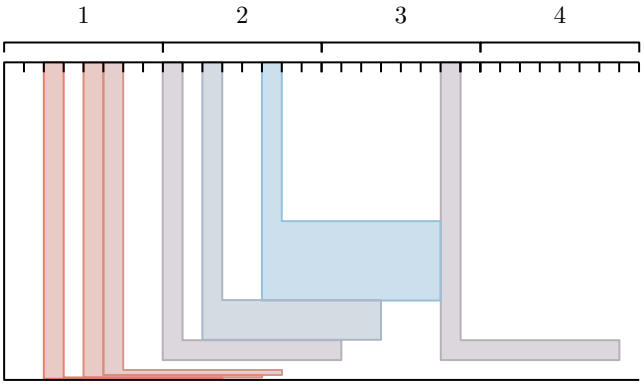


Figure 6 Illustration for the proof of Lemma 12.

We now argue that this yields a valid bin packing. Let L_i and L_j be two L-shapes in a same bin and denote their x-coordinates in the strip by x_i and x_j, and in the bin by x'_i and x'_j, respectively. We assume that $x_i < x_j$. By construction, we have that $|x_i - x_j| \leq 1/2$. The placement $x_i < x_j$ implies that $w_y(L_i) + \ell_y(L_j) \leq 1$ and hence $i \geq j$, i.e., in each bin the L-shapes are non-increasing. Moreover, due to the equal width of $1/2n$, no two L-shapes intersect in the x-direction. Without duplicates, no intersections in the y-direction follow from the definition of $\mathcal{L}^n$. Lastly, if $x'_j - x'_i \leq {}^{n-1}/_{2n}$ then $x'_j + \ell_x(L_i) \leq ({}^{n-1}/_{2n} + x_i) + 1/2 + 1/2k \leq x_i + 1$. Therefore, the y-span of the L-shapes is at most 1.

It remains to check the number of bins. Let OPT_S and OPT_B denote the offline optimum values for L-StripPack and L-BinPack, respectively. Clearly, we have $\mathrm{OPT}_S \leq \mathrm{OPT}_B = 1$; recall that all objects pack into one bin by Lemma 9. As $\mathcal{A}$ is asymptotically α-competitive, there is a constant $\beta > 0$ such that the width of the strip packing is at most $\alpha \cdot \mathrm{OPT}_S + \beta$. The number of bins is $\lceil \lceil 2n(\alpha \cdot \mathrm{OPT}_S + \beta) \rceil / n \rceil \leq 2\alpha \mathrm{OPT}_B + 2\beta + 2$, so $\mathcal{A}'$ is asymptotically 2α-competitive. ◀

Together, Lemma 12 and Corollary 11 imply Corollary 4(i).

5 Orthogonal polygons of higher complexity – Theorems 5 and 6(ii)

As a stepping stone towards a proof of Theorem 5, we first consider Z-skeletons.

▶ **Proposition** (corresponding to Theorem 6(ii)). *No online algorithm for Z-Skel-BinPack has an asymptotic competitive ratio better than n.*

For the proof, we consider an arbitrary online algorithm $\mathcal{A}$ for Z-Skel-BinPack, and use Algorithm 1 to generate a sequence of Z-skeletons such that they can be packed into one bin, but $\mathcal{A}$ is forced to use one bin per Z-skeleton.

Algorithm 1 Strategy to generate a sequence of Z-skeletons for a given algorithm $\mathcal{A}$.

function `GenerateZSkeletons`$(n, \mathcal{A})$
  $b \leftarrow 0, B \leftarrow 1$
  for each $i \in [n]$ **do**
    Present Z_i with $w_i := 1 - 2^{-i-1}$, $b_i := (b + B)/2$ and $a_i := 1 - b_i$.
    $x_i \leftarrow x$-coordinate of Z_i assigned by $\mathcal{A}$
    if $x_i \leq (1 - w_i)/2$ **then** $B \leftarrow b_i$ **else** $b \leftarrow b_i$

▶ **Lemma 13.** *For every online algorithm $\mathcal{A}$ of Z-Skel-BinPack, Algorithm 1 generates a sequence of n Z-skeletons such that $\mathcal{A}$ uses n bins.*

Proof Sketch. For any Z-skeleton Z_i packed by $\mathcal{A}$, Algorithm 1 ensures that all future Z_j it presents cannot be placed into the bin of Z_i. In particular, if $\mathcal{A}$ places Z_i far to the left, all $Z_j, j > i$, have a longer upper arm than Z_i, but are too wide to fit under Z_i, as in Figure 7a. Similarly, if $\mathcal{A}$ places a Z-skeleton far to the right, then all $Z_j, j > i$, have a shorter upper arm than Z_i, but are too wide to fit above Z_i, as in Figure 7b. ◀



(a) (b)

Figure 7 Illustration for the proof of Lemma 13.

In order to expand the Z-skeletons to Z-shapes, we ensure the Z-skeletons can even be packed with some positive distance to each other.

▶ **Lemma 14.** *Let $Z_1, \ldots, Z_n$ be Z-skeletons generated by some execution of Algorithm 1. Then, $Z_1, \ldots, Z_n$ can be packed into one bin, such that all horizontal gaps between the Z-skeletons and the bin boundary have width at least $1/n \cdot 2^{-n-3}$.*

Proof Sketch. First, we show that we may partition the generated Z-skeletons into two subsets $W^{\uparrow}$ and $W^{\downarrow}$, where the parameter w_k grows or shrinks, respectively, as the parameter b_k grows. By induction on k, we show that, given any $\varepsilon > 0$, the shapes of $W^{\uparrow}$ can be packed into the L-shape with parameters $\varepsilon + w_k, \varepsilon, 1, b_k$, depicted in Figure 8a, such that all horizontal gaps have size ε/k.

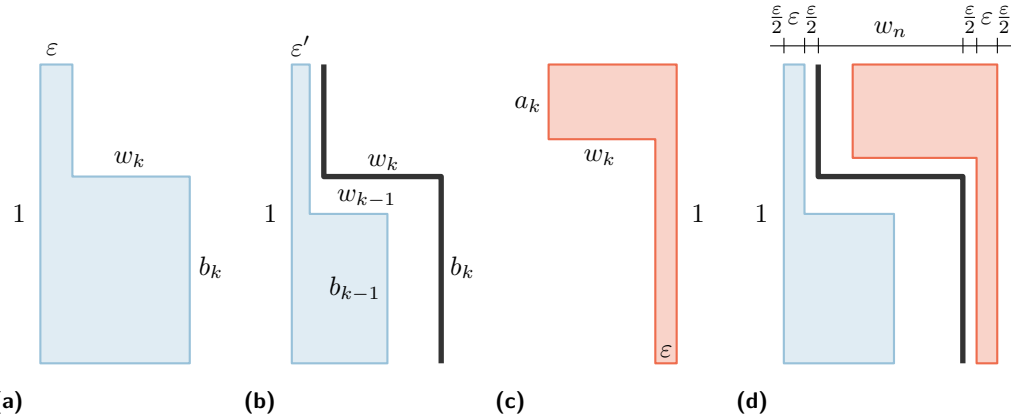


Figure 8 Illustration for Lemma 14.

Similarly, $W^{\downarrow}$ can be packed into a rotated L-shape with parameters $\varepsilon + w_k, \varepsilon, 1, a_k$, illustrated in Figure 8c. Finally, both of these L-shapes can be packed into a unit bin with the desired distances, see Figure 8d. ◀

Lemmas 13 and 14 directly imply Theorem 6(ii). Moreover, the horizontal gap sizes of Lemma 14 allow to expand the Z-skeletons of Algorithm 1 to Z-shapes with thickness $1/n \cdot 2^{-n-3}$. Lemma 14 guarantees that if the Z-skeletons cannot be packed disjointly, the Z-shapes cannot be packed interiorly disjoint. Consequently, there is no online algorithm for Ortho-BinPack of Z-shapes of equal thickness that achieves an asymptotic competitive ratio better than n, which implies the following result.

▶ **Theorem 5.** *There is no online algorithm for Ortho-BinPack that achieves an asymptotic competitive ratio better than n, even when restricting to orthogonally convex 8-gons.*

The similarity to Lemma 13 is in contrast to orthogonal 6-gons, where bounds on the competitive ratios for L-skeletons and L-shapes differ drastically, cf. Theorems 2 and 6(i).

6 Conclusion and open problems

In this work, we investigated various packing problems of orthogonal polygons under translation. For L-shapes, the yet smallest open case, we wondered whether they behave more like rectangles or convex polygons. For bin packing, we provided a surprisingly large lower bound on the competitive ratio of about $n/\log n$ for general L-shapes and n for general orthogonal polygons. That means that the trivial algorithm, which packs each item into its private bin, is worst-case optimal for orthogonal polygons. These lower bounds are much higher than for convex polygons, cf. Table 2. In contrast, for symmetric and small L-shapes there exist constant competitive algorithms. This insight implies that in terms of the competitive ratio, L-shapes behave more like rectangles for perimeter packing, area packing, and the online critical density.

Many interesting problems remain for future work. Most notably, it remains to improve the bounds or even close the gaps for the problems stated in Tables 1 and 2. This includes the problem of determining the best competitive ratio for online bin packing of L-shapes, as it remains open whether it it possible to beat the trivial algorithm for L-BinPack.

References

1 Anders Aamand, Mikkel Abrahamsen, Lorenzo Beretta, and Linda Kleist. Online sorting and translational packing of convex polygons. In *Symposium on Discrete Algorithms (SODA 2023)*, pages 1806–1833. SIAM, 2023. doi:10.1137/1.9781611977554.ch69.

2 Mikkel Abrahamsen, Ioana O. Bercea, Lorenzo Beretta, Jonas Klausen, and László Kozma. Online Sorting and Online TSP: Randomized, Stochastic, and High-Dimensional. In *European Symposium on Algorithms (ESA 2024)*, volume 308 of *LIPIcs*, pages 5:1–5:15, 2024. doi:10.4230/LIPIcs.ESA.2024.5.

3 Mikkel Abrahamsen and Lorenzo Beretta. Online Packing to Minimize Area or Perimeter. In *37th International Symposium on Computational Geometry (SoCG 2021)*, pages 6:1–6:15, 2021. doi:10.4230/LIPIcs.SoCG.2021.6.

4 Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. Framework for $\exists\mathbb{R}$-completeness of two-dimensional packing problems. *TheoretiCS*, 3, 2024. doi:10.46298/theoretics.24.11.

5 Helmut Alt. Computational aspects of packing problems. *Bulletin of the EATCS*, 118, 2016. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/390.

6 Helmut Alt, Otfried Cheong, Ji-won Park, and Nadja Scharf. Packing 2D disks into a 3D container. In *International Workshop on Algorithms and Computation (WALCOM 2019)*, pages 369–380, 2019. doi:10.1007/978-3-030-10564-8_29.

7 Helmut Alt, Mark de Berg, and Christian Knauer. Approximating minimum-area rectangular and convex containers for packing convex polygons. *Journal of Computational Geometry*, 8(1):1–10, 2017. doi:10.20382/jocg.v8i1a1.

8 Helmut Alt, Mark de Berg, and Christian Knauer. Corrigendum to: Approximating minimum-area rectangular and convex containers for packing convex polygons. *Journal of Computational Geometry*, 11(1):653–655, 2020. doi:10.20382/jocg.v11i1a26.

9 Yossi Azar, Debmalya Panigrahi, and Or Vardi. Nearly tight bounds for the online sorting problem. *CoRR*, abs/2508.14287, 2025. doi:10.48550/arXiv.2508.14287.

10 Brenda S. Baker and Jerald S. Schwarz. Shelf algorithms for two-dimensional packing problems. *SIAM Journal on Computing*, 12(3):508–525, 1983. doi:10.1137/0212033.

11 János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. Lower bounds for several online variants of bin packing. *Theory of Computing Systems*, 63:1757–1780, 2019. doi:10.1007/s00224-019-09915-1.

12 János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. A new lower bound for classic online bin packing. *Algorithmica*, 83:2047–2062, 2021. doi:10.1007/s00453-021-00818-7.

13 János Balogh, Leah Epstein, and Asaf Levin. More on ordered open end bin packing, 2020.

14 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *Symposium on discrete algorithms (SODA 2014)*, pages 13–25. SIAM, 2014. doi:10.1137/1.9781611973402.2.

15 Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM*, 54(1):1–39, 2007. doi:10.1145/1206035.120603.

16 Brian Brubach. Improved bound for online square-into-square packing. In *International Workshop on Approximation and Online Algorithms (WAOA 2014)*, pages 47–58, 2014. doi:10.1007/978-3-319-18263-6_5.

17 Lin Chen, Nicole Megow, and Kevin Schewior. The Power of Migration in Online Machine Minimization. In *Symposium on Parallelism in Algorithms (SPAA 2016) and Architectures*, pages 175–184, Pacific Grove California USA, 2016. doi:10.1145/2935764.2935786.

18 Lin Chen, Nicole Megow, and Kevin Schewior. An $O(\log m)$-Competitive Algorithm for Online Machine Minimization. *SIAM Journal on Computing*, 47(6):2057–2077, 2018. doi:10.1137/17M116032X.

19 Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017. doi:10.1016/j.cosrev.2016.12.001.

20 Don Coppersmith and Prabhakar Raghavan. Multidimensional On-Line Bin Packing: Algorithms and Worst-Case Analysis. *Operations Research Letters*, 8(1), 1989. doi:10.1016/0167-6377(89)90027-8.

21 János Csirik and Gerhard J. Woeginger. On-line packing and covering problems. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 147–177. Springer, 1998. doi:10.1007/BFb0029568.

22 Nikhil Devanur, Konstantin Makarychev, Debmalya Panigrahi, and Grigory Yaroslavtsev. Online Algorithms for Machine Minimization, 2014. doi:10.48550/arXiv.1403.0486.

23 Leah Epstein. A lower bound for online rectangle packing. *Journal of Combinatorial Optimization*, 38:846–866, 2019. doi:10.1007/s10878-019-00423-z.

24 Leah Epstein. Open-end bin packing: new and old analysis approaches. *Discrete Applied Mathematics*, 321:220–239, 2022. doi:10.1016/j.dam.2022.07.003.

25 Leah Epstein and Loay Mualem. Online bin packing of squares and cubes. *Algorithmica*, 85(5):1415–1458, 2023. doi:10.1007/S00453-022-01078-9.

26 Leah Epstein and Rob van Stee. Multidimensional packing problems. In Teofilo F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, pages 553–570. Chapman and Hall/CRC, second edition, 2018. doi:10.1201/9781351236423-31.

27 Sándor P. Fekete and Hella-Franziska Hoffmann. Online square-into-square packing. *Algorithmica*, 77(3):867–901, 2017. doi:10.1007/s00453-016-0114-2.

28 Robert Fowler, Michael Paterson, and Steven Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information processing letters*, 12(3):133–137, 1981. doi:10.1016/0020-0190(81)90111-3.

29 Gábor Galambos. A 1.6 lower-bound for the two-dimensional on-line rectange bin-packing. *Acta Cybernetica*, 10(1-2):21–24, January 1991. URL: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3389.

30 Tim Gerlach, Benhamin Hennies, and Linda Kleist. Online packing of orthogonal polygons, 2026. doi:10.48550/arXiv.2603.22098.

31 Xin Han, Francis Y. L. Chin, Hing-Fung Ting, Guochuan Zhang, and Yong Zhang. A new upper bound 2.5545 on 2d online bin packing. *ACM Transactions on Algorithms*, 7(4):50:1–50:18, 2011. doi:10.1145/2000807.2000818.

32 Xin Han, Kazuo Iwama, Deshi Ye, and Guochuan Zhang. Strip packing vs. bin packing. In *Algorithmic Aspects in Information and Management (AAIM 2007)*, pages 358–367, 2007. doi:10.1007/978-3-540-72870-2_34.

33 Xin Han, Kazuo Iwama, and Guochuan Zhang. Online removable square packing. *Theory of Computing Systems*, 43(1):38–55, 2008. doi:10.1007/s00224-007-9039-0.

34 Rolf Harren, Klaus Jansen, Lars Prädel, Ulrich M. Schwarz, and Rob van Stee. Two for one: Tight approximation of 2D bin packing. *International Journal of Foundations of Computer Science*, 24(8):1299–1328, 2013. doi:10.1142/S0129054113500354.

35 Rolf Harren, Klaus Jansen, Lars Prädel, and Rob Van Stee. A $(5/3+\varepsilon)$-approximation for strip packing. *Computational Geometry*, 47(2):248–267, 2014. doi:10.1016/j.comgeo.2013.08.008.

36 Sungjin Im, Benjamin Moseley, Kirk Pruhs, and Clifford Stein. An $O(\log\log m)$-Competitive Algorithm for Online Machine Minimization, 2018. doi:10.48550/arXiv.1708.09046.

37 Janusz Januszewski and Marek Lassak. On-line packing sequences of cubes in the unit cube. *Geometriae Dedicata*, 67(3):285–293, 1997. doi:10.1023/A:1004953109743.

38 Mong-Jen Kao, Jian-Jia Chen, Ignaz Rutter, and Dorothea Wagner. Competitive design and analysis for machine-minimizing job scheduling problem. In *International Symposium on Algorithms and Computation*, pages 75–84, 2012. doi:10.1007/978-3-642-35261-4_11.

39 Henry A. Kierstead and William T. Trotter. An extremal problem in recursive combinatorics. *Congressus Numerantium*, 33(143-153):98, 1981.

40 A. Kosiński. A proof of an Auerbach-Banach-Mazur-Ulam theorem on convex bodies. *Colloquium Mathematicae*, 4(2):216–218, 1957. doi:10.4064/cm-4-2-216-218.

41 Adam Kurpisz and Silvan Suter. Improved Approximations for Translational Packing of Convex Polygons. In *European Symposium on Algorithms (ESA 2023)*, volume 274 of *LIPIcs*, pages 76:1–76:14, 2023. doi:10.4230/LIPIcs.ESA.2023.76.

42 Marek Lassak and Jixian Zhang. An on-line potato-sack theorem. *Discrete & Computational Geometry*, 6(1):1–7, 1991. doi:10.1007/BF02574670.

43 Joseph Leung, Tommy Tam, Chin Wong, Gilbert Young, and Francis Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990. doi:10.1016/0743-7315(90)90019-L.

44 R. Daniel Mauldin. The scottish book: Mathematics from the scottish café, with selected problems from the new scottish book, 2015. doi:10.1007/978-3-319-22897-6.

45 Arturo I. Merino and Andreas Wiese. On the two-dimensional knapsack problem for convex polygons. In *International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, LIPIcs, pages 84:1–84:16, 2020. doi:10.4230/LIPIcs.ICALP.2020.84.

46 J.W. Moon and L. Moser. Some packing and covering theorems. In *Colloquium Mathematicum*, volume 17, pages 103–110, 1967. doi:10.4064/cm-17-1-103-110.

47 Jubayer Nirjhor and Nicole Wein. Improved online sorting. In *Approximation and Online Algorithms (WAOA 2025)*, pages 187–197, 2025. doi:10.1007/978-3-032-06706-7_13.

48 Barna Saha. Renting a Cloud. In *Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013)*, volume 24 of *LIPIcs*, pages 437–448, 2013. doi:10.4230/LIPIcs.FSTTCS.2013.437.

49 Rob van Stee. SIGACT news online algorithms column 20: the power of harmony. *SIGACT News*, 43(2):127–136, 2012. doi:10.1145/2261417.2261440.

50 Rob van Stee. SIGACT news online algorithms column 26: Bin packing in multiple dimensions. *SIGACT News*, 46(2):105–112, 2015. doi:10.1145/2789149.2789167.

51 André van Vliet. An improved lower bound for on-line bin packing algorithms. *Information processing letters*, 43(5):277–284, 1992. doi:10.1016/0020-0190(92)90223-I.

52 Deshi Ye, Xin Han, and Guochuan Zhang. A note on online strip packing. *Journal of Combinatorial Optimization*, 17(4):417–423, 2009. doi:10.1007/s10878-007-9125-x.

53 Guosong Yu and Guochuan Zhang. Scheduling with a minimum number of machines. *Operations Research Letters*, 37(2):97–101, 2009. doi:10.1016/j.orl.2009.01.008.

Constructing Doppelgängers of Greedy Geometric Spanners in Practice

Anirban Ghosh
University of North Florida, Jacksonville, FL, USA

Abstract

Greedy geometric spanners are considered to be the gold standard for their near-optimal guarantees in terms of sparsity and total weight. However, their inefficient construction poses significant challenges for large-scale geometric networks, especially for low values of stretch factors (< 2). We present Θ-Greedy, a simple and practical parallel algorithm engineered for constructing *doppelgängers* of greedy geometric spanners that empirically resemble the greedy spanners in key structural and performance metrics, including average degree, degree, and lightness. Unlike approximate greedy spanners, *doppelgängers* of greedy spanners are almost indistinguishable from the actual greedy spanners in practice.

In our experiments, Θ-Greedy consistently produced greedy spanner doppelgängers across a broad range of synthetic and real-world datasets, offering the first practical alternative to the computationally intensive greedy spanners. Θ-Greedy can construct a 1.1-spanner on a $128K$-element uniformly distributed point set in well under 5 minutes. In contrast, Bucketing, the most practical greedy spanner algorithm, takes around 3 hours. For million-sized point sets, Θ-Greedy can run to completion in a few hours, making it much faster than Bucketing, which takes days to finish. In extensive experiments on synthetic and real-world datasets, Θ-Greedy delivered speedups of up to 147x over Bucketing while preserving greedy-like sparsity and weight.

For broader uses of the algorithm and reproducibility, we share our engineered C++ code.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases geometric graph, geometric spanners, greedy spanners, algorithm engineering

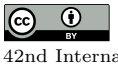
Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.53

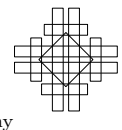
Supplementary Material *Software (Source Code)*: `https://doi.org/10.5281/zenodo.19238942`

Acknowledgements The author would like to thank the anonymous reviewers for their insightful comments and suggestions, which helped to significantly improve the clarity and quality of this paper.

1 Introduction

A *geometric spanner* on a set of points P in L_2 metric is a sparse subgraph of the complete Euclidean graph on P that approximately preserves pairwise distances among all point pairs in P. Formally, given a set of n points $P \subset \mathbb{R}^d$ and a *stretch factor* $t \geq 1$, a geometric graph $H := (P, E)$ is a t-spanner if, for all $u, v \in P$, the shortest-path distance in H, denoted by $\delta_H(u, v)$, satisfies $\delta_H(u, v) \leq t \cdot |uv|$, where $|uv|$ denotes the Euclidean distance between u, v. A t-spanner is also a t'-spanner, for all $t' \geq t$. A t-path exists between two points in a geometric graph H if $\delta_H(u, v) \leq t \cdot |uv|$. Therefore, H is a t-spanner if for every point pair in P, a t-path exists in H. The *average degree* of H is defined as $2|E|/|n|$, and its *lightness* is the ratio of the weight of H to the weight of a Euclidean minimum spanning tree of P. For a fixed value of t, lower average degree and lightness are better. The complete Euclidean graph is a 1-spanner with $\binom{n}{2} = \Theta(n^2)$ edges, rendering it impractical for large-scale applications. So, the study of geometric spanners focuses on designing algorithms that construct sparse t-spanners, ideally with $O(n)$ edges, while maintaining desirable geometric and algorithmic properties; see [13, 28, 34] for a comprehensive overview. Spanners play a central role in


© Anirban Ghosh;
licensed under Creative Commons License CC-BY 4.0




computational geometry and the design of geometric networks. They are widely used in applications such as wireless networks, motion planning, and approximation algorithms, where maintaining low stretch with minimal edge cost is essential [5, 18, 24, 27, 30, 31].

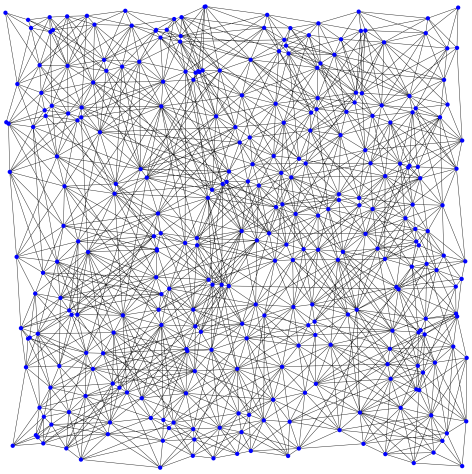
Introduced by Althöfer et al. [4] and also by Chandra et al. [16] more than 30 years ago, the greedy spanner is a cornerstone in the study of geometric spanner networks, offering near-optimal guarantees in terms of sparsity and weight (also lightness) [10, 20, 21, 26, 32]. The greedy spanner is naively constructed by processing all pairs of points in order of increasing Euclidean distance and adding an edge between each pair (u, v) if the current shortest path between them exceeds $t \cdot |uv|$, where $t \geq 1$ is the desired stretch factor. The algorithm, known as Path-Greedy in the literature, is a generalization of the textbook Kruskal's algorithm. Refer to Figure 1 for greedy spanner examples. A straightforward implementation of Path-Greedy runs in $O(n^3 \log n)$ time since there are $\binom{n}{2}$ point pairs, and for every pair, computation of a shortest path takes $O(n \log n)$ time, as greedy spanners are linear-sized spanners. It also uses $\Theta(n^2)$ space to store and sort all point pairs, making it impractical for large point sets. A variant (still using $\Theta(n^2)$ extra space), named FG-Greedy, was introduced by Farshi and Gudmundsson [22], to speed up the algorithm by reducing the number of shortest-path calls. The idea was to memorize the shortest-path lengths obtained by previous single-source shortest-path calls using a matrix, allowing future calls to be avoided using simple lookups. Surprisingly, it was shown in [11] that FG-Greedy runs in $O(n^3 \log n)$ time, although in experiments, it was found to be faster than Path-Greedy; refer to [22]. Das and Narasimhan made the first attempt to approximate greedy spanners in [19], wherein they designed the Approximate-Greedy algorithm that runs in $O(n \log^2 n)$ time. See Table 1. Despite being theoretically novel, experimental studies [22, 32] on spanners found that Approximate-Greedy places substantially more edges compared to actual greedy spanners, rendering them impractical for real-world applications. Bose et al. [12] designed an $O(n^2 \log n)$-time algorithm to construct exact greedy spanners using $\Theta(n^2)$ space, an improvement over Path-Greedy. To see how frustrating quadratic space algorithms are in practice, consider $n = 10^6$. Storing all point pairs, at least $\binom{10^6}{2} \cdot 8$ `bytes` $= 4$ `TB` of space is required, which is impractical. For this reason, we chose not to consider it in our experiments.

Table 1 A comparison of the popular spanner construction algorithms vs Θ-Greedy (ours), based on the average degree (Δ_{avg}) of the generated spanners. The input was a $128K$-element uniformly distributed point set, and $t = 1.1$.

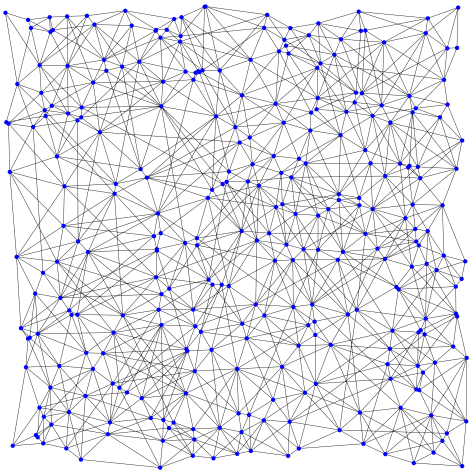
	Θ-Graph [17]	Sink-Spanner [8]	WSPD-Spanner [14]	Skip-List-Spanner [7]
Δ_{avg}	97.38	123.13	7646.93	192.44

	Appx-Greedy [19]	Fast-Sparse-Spanner [32]	Bucketing [3]	Θ-Greedy
Δ_{avg}	583.40	9.18	9.08	9.08

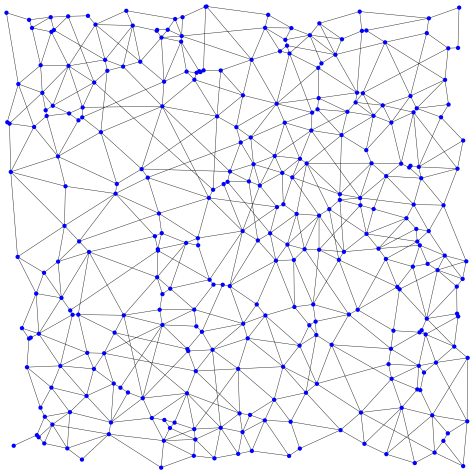
Alewijnse et al. [2] made the first significant attempt to construct greedy spanners on large point sets, marking a breakthrough. Their algorithm, referred to as WSPD-Greedy in [3], runs in $O(n^2 \log^2 n)$ time using only $O(n)$ space, enabling the construction of geometric spanners on million-point instances for the first time. It avoids storing all pairwise distances by incrementally generating edge candidates via a well-separated pair decomposition (WSPD) [15, 25]. These candidates are processed in order of increasing distance using a priority queue, ensuring that only necessary edges are added while maintaining linear space complexity. The same authors devised an improved algorithm in [3], named Bucketing. It is a distribution-sensitive algorithm for constructing greedy spanners that exploits the geometric structure of point sets. They observed that most short edges can be determined locally, while



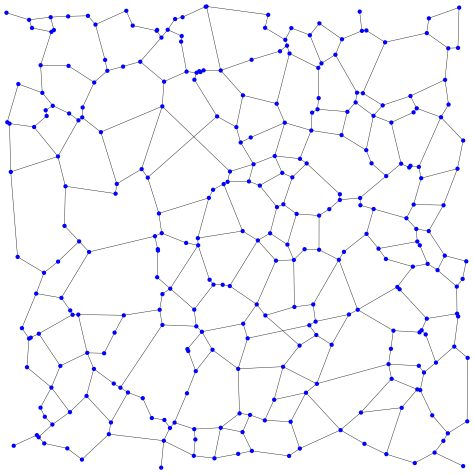
(a) $t = 1.05, |E| = 1743$



(b) $t = 1.1, |E| = 1211$



(a) $t = 1.25, |E| = 735$



(b) $t = 2, |E| = 408$

Figure 1 Greedy spanners constructed on a 300-element uniformly distributed point set. The complete graph on the point set has $\binom{300}{2} = 44850$ edges.

long edges can be efficiently identified using the well-separated pair decomposition (WSPD). By leveraging a local characterization of t-spanners, Bucketing achieves subquadratic expected time of $O(n \log^2 n \log^2 \log n)$ on uniform distributions, making it the first such result for greedy spanners on any meaningful class of inputs. Bucketing was observed to be much faster than WSPD-Greedy; see [3] for experimental results. Abu-Affash et al. [1] designed the δ-greedy algorithm that retains the desirable properties of the Path-Greedy spanner, such as low degree, linear number of edges, and near-optimal weight, while significantly reducing the number of shortest path queries. The algorithm runs in $O(n^2 \log n)$ time in the worst case, but achieves expected $O(n \log n)$ time on uniformly random point

sets. The parameter δ controls the trade-off between construction time and similarity to the Path-Greedy spanner; setting $\delta = t$ yields an identical spanner. However, the space complexity is quadratic, making it impractical for large point sets. Therefore, we decided not to include it in our experiments.

Shariful et al. [32] recently observed that Bucketing is slow for large values of n. For instance, when $n = 10^6$, $t = 1.1$, and the points are uniformly distributed inside a square, Bucketing took around five days to generate the spanner. So, they engineered a practical parallel algorithm Fast-Sparse-Spanner, that generated geometric graphs fast and had average degrees close to those of greedy spanners. Fast-Sparse-Spanner could generate a 1.1-spanner for the same above point set in just 55 minutes, making it much faster than Bucketing in practice. They experimentally observed that Fast-Sparse-Spanner consistently generated t-spanners across a wide variety of distributions and real-world point sets. To check for the t-spanner property, they also devised an auxiliary practical stretch factor measurement algorithm, named Fast-Stretch-Factor, since Fast-Sparse-Spanner is not theoretically guaranteed to generate t-spanners. However, the weights and degrees of the generated graphs are far from the actual greedy spanners owing to the presence of long WSPD spanner edges, which form a backbone of their spanners. Despite the average degree of the generated graphs being close to those generated by Bucketing (see Table 1), the degrees and lightness values substantially differed. For the point set used in Table 1, the degrees of graphs generated by Fast-Sparse-Spanner and Bucketing were 78 and 18, respectively, and the lightness values were 16.7196 and 11.0354, respectively.

Despite the progress made by Shariful et al. in [32], practical construction of spanners which closely resemble greedy spanners in terms of average degree, weight, and degree, especially for low t-values (≤ 2), remains an open problem. Note that Delaunay triangulations are 1.998-spanners [35] with $O(n)$ edges that can be constructed in $O(n \log n)$ time (in practice, takes a few seconds even for millions of points). So, constructing t-spanners with t around 2 or more is not considered challenging in practice.

What are doppelgängers of greedy spanners? We introduce the notion of (a, b)-*doppelgänger*. A t-spanner H on a point set P is a (a, b)-*doppelgänger* of the greedy spanner G on P, if its average degree and lightness do not exceed those of G by more than additive constants a and b, respectively. Average degree and lightness are two central measures of spanner quality, and spanners with similar values for these metrics tend to be structurally similar. Thus, smaller values of a and b indicate better doppelgängers. Such doppelgängers are especially appealing to practitioners when they can be constructed significantly faster than the greedy spanner itself. The term "approximation" has not been used in this work intentionally, as the Approximate-Greedy [19] algorithm claims to generate spanners that are approximations of greedy spanners, but, in reality, they are structurally very different from the actual greedy spanners (also, see Table 1).

Our contribution. Motivated by the lack of fast practical algorithm to generate (a, b)-*doppelgängers* of greedy spanners for low values of a and b, we design and engineer Θ-Greedy, a new spanner construction algorithm, based on Θ-Graph. We found Θ-Greedy to be considerably faster than our closest competitor Bucketing, and always produced high quality *doppelgängers* in our experiments. In our experiments, both a and b never exceeded 0.1. To our knowledge, no other known algorithm can achieve this while being fast and scalable in practice. See Figure 2 for a visual comparison between Θ-Greedy and Bucketing.

We use the Θ-Graph as a seed spanner since it can be constructed fast in $O(n \log n)$ time and, as seen in Table 1, excluding Fast-Sparse-Spanner (not guaranteed to produce t-spanners), Θ-Graphs have the lowest average degree and thus the lowest number of edges. We consider its edges by length and place them greedily, akin to Path-Greedy. Finally, we find point pairs that do not have t-paths between them using a quad tree-based approach. For every such point pair, we establish a t-path by placing edges carefully. Our algorithm can leverage multiple cores for speedup.

Θ-Greedy is designed as an engineering solution to a long-standing practical bottleneck in geometric spanner construction rather than as a theoretical breakthrough. While its worst-case complexity is $O(n^3 \log n)$, the algorithm consistently exhibits near-quadratic scaling in practice and uses linear space, enabling it to handle million-point datasets efficiently. It leverages heuristic path checks and shortcut caching to achieve massive speedups, up to $\approx$ 147x faster than Bucketing, while producing spanners that are structurally almost indistinguishable from true greedy spanners in terms of degree and lightness. The focus of Θ-Greedy is on simplicity, scalability, and reproducibility for real-world geometric networks, not on improving asymptotic bounds.

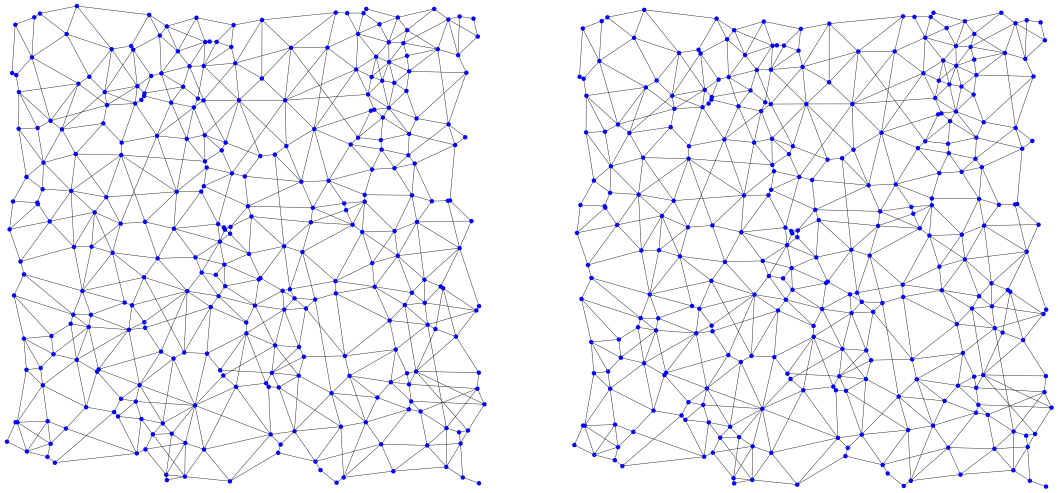


Figure 2 $n = 300, t = 1.25$; the spanners produced by Bucketing (Left) and Θ-Greedy (Right).

Preliminaries. A *region quad tree* is a hierarchical spatial data structure that recursively partitions a two-dimensional region into four congruent quadrants until each subregion is homogeneous or its size falls below a specified threshold ℓ, the maximum leaf size. The root node represents the entire region, and each internal node corresponds to a square subdivided into four child quadrants. A Θ-Graph [28] is a geometric spanner constructed by dividing the plane around each point into k equal-angle cones and connecting the point to the vertex in each cone that minimizes the distance projected onto the cone's bisector. The constructed geometric graph is a t-spanner if $k = \max\left(9, \frac{2\pi}{\Theta}\right)$, where $\Theta = \frac{\arcsin\left(1-\frac{1}{t^2}\right)}{2}$. Θ-Graph can be constructed in $O(n \log n)$ time [28].

2 Description of Θ-Greedy

This section presents the steps of the Θ-Greedy algorithm in order. Let H denote the output spanner constructed on the input point set P of size n. Initially, H is an empty graph.

Step 1. Construct a minimum spanning tree M on P by running Prim's algorithm on the edges of a Delaunay triangulation of P. Place the $n-1$ edges of M in H. Since the Path-Greedy algorithm will place such edges, irrespective of the value of t, we can compute such edges quickly and place them in advance to avoid shortest-path calls.

Step 2. Refer to Algorithm 1. We construct a Θ-Graph t-spanner G_Θ on P. Let E_Θ be the set of edges in G_Θ. Sort the edges in E_Θ in non-decreasing order of length. Then, process them greedily, following the spirit of the Path-Greedy algorithm: For each edge $e = \{u, v\} \in E_\Theta$, if no t-path exists between u and v in H, insert $\{u, v\}$ into H; otherwise, skip the edge.

To check for an t-path between u, v, we first invoke an heuristic algorithm called Greedy-Path, which avoids the use of a priority queue, unlike standard shortest-path algorithms such as A^* or Dijkstra. This heuristic is inspired by the work [32]. We make use of `shortcuts`, as described next, to speed it up. The Greedy-Path performs a walk from u to v in H, aiming to discover a path greedily, whose total length does not exceed $t \cdot |uv|$. At each step, it evaluates candidate next hops drawn from two sources: (i) precomputed t-paths obtained by previous Greedy-Path and A^* calls, stored in `shortcuts`, as described next (also see Algorithm 1), and (ii) the actual graph neighbors of the current vertex. For every candidate, it computes a simple score - the current path length greedily constructed so far plus the edge (or shortcut) cost to that candidate, plus its straight-line (Euclidean) distance to v as an optimistic estimate of continuing via that vertex. It then moves to the candidate with the lowest score, marking visited choices to avoid cycling, and it stops immediately if v is reached. If the accumulated path length exceeds the stretch upper bound, the heuristic declares failure; otherwise, it returns the achieved t-path.

If Greedy-Path finds a t-path, we store it as a shortcut (with its weight) in a structure `shortcuts`, similar to an adjacency list, to accelerate future queries. If Greedy-Path fails, we fall back to the popular A^* algorithm. If A^* also fails, we insert u, v into H; otherwise, we store the discovered t-path in `shortcuts` for future use. Note that to maintain the linear size of `shortcuts`, it is not updated in Steps 3, 4, and 5 of our algorithm.

Step 3. Construct a region quad tree T on P such that the number of points contained inside every leaf is no more than a fixed constant ℓ. In our experiments, we found that $\ell = 1000$ was sufficient. For every leaf $\sigma \in T$, ensure that a t-path exists in H_σ, between all point pairs in σ, where H_σ is a subgraph of H, induced by the points in σ. This is achieved by running Dijkstra from every point in σ. Let S be the set of point pairs from all the leaves that do not have t-paths in H. Sort S, and place edges greedily. Akin to Step 2, we first use Greedy-Path, followed by A^*, if necessary. While adding edges greedily, we check for the existence of t-paths by considering H rather than H_σ.

Step 4. Let E' be an empty set of edges. For every leaf pair $\sigma_i, \sigma_j \in T$, ensure t-paths in H between all point pairs $u \in \sigma_i, v \in \sigma_j$. If a pair $\{u, v\}$ is found not to have a t-path in H, place the edge $\{u, v\}$ in H and add the same to E'. Akin to Step 2, we use Greedy-Path followed by A^*. For a given leaf pair σ_i, σ_j, the t-paths (bridges connecting distant point pairs) found are temporarily stored in a list $\mathcal{B}$. Since $|\sigma_i| \cdot |\sigma_j| \leq \ell^2 = O(1)$, $|\mathcal{B}| = O(1)$. Its use is based on the observation that points from faraway tend to share t-paths. Consequently, we save

Algorithm 1 STEP 2.

```
Construct a Θ-GRAPH t-spanner G_Θ = (P, E_Θ);
Sort E_Θ based on edge lengths;
for every edge {u, v} in E_Θ do
    Let l_{u,v} be the length of the path computed by GREEDY-PATH(H, u, v);
    if l_{u,v} ≤ t · |uv| then
        Store the triplets (u, v, l_{u,v}) and (v, u, l_{u,v}) in shortcuts;
    else
        Let δ_H(u, v) be the length of the shortest-path computed by A*(H, u, v);
        if δ_H(u, v) ≤ t · |uv| then
            Store the triplets (u, v, δ_H(u, v)) and (v, u, δ_H(u, v)) in shortcuts;
        else
            Place the edge {u, v} in H;
        end if
    end if
end for
```

t-path computations by first searching for a valid bridge (x, y) in $\mathcal{B}$, where $x \in \sigma_i, y \in \sigma_j$ such that $t \cdot |ux| + |xy| + t \cdot |yv| \leq t \cdot |uv|$. If $|E'| > n$, terminate this step early and go to Step 5. Since every leaf pair is considered independently, a super-linear number of edges could be placed in this step. The following step is a fail-safe step to ensure linearity.

Step 5. If $|E'| > n$, delete all the edges in E' from H. For every leaf pair $\sigma_i, \sigma_j \in T$, find all point pairs $u \in \sigma_i, v \in \sigma_j$, that do not have t-path in H. Similar to Step 4, GREEDY-PATH is used first, followed by A^* (if the former fails to find a t-path). Let X be the set of all such pairs. Sort X, and place edges greedily in H.

Next, we show in Theorem 1 that H is a sparse t-spanner on P.

▶ **Theorem 1.** *For any fixed $t > 1$, H is a t-spanner with $O(n)$ edges.*

Proof. The algorithm begins by inserting the edges of a minimum spanning tree M into H. Since M is connected and contains exactly $n - 1$ edges, H is connected after Step 1. In Step 2, we place $O(n)$ edges since the size of G_Θ is $O(n)$, for fixed t. Step 3 introduces $O(n)$ edges as ℓ is a constant. We insert at most $\binom{\ell}{2} = O(1)$ edges within each leaf of T, and since T has $O(n)$ leaves, the total number of edges added is $O(n)$. If Step 4 places at most n edges, then clearly, H has $O(n)$ edges.

Now, let us assume that the number of edges placed in Step 4 exceeds n (possibly $\omega(n)$). In what follows, we argue that in such a scenario, Step 5 places $O(n)$ edges, thereby proving our sparsity claim for H.

Let u be any vertex of H. We claim that the number of edges placed in Step 5, that are incident on u, is at most $\lceil 2\pi/\alpha \rceil$, such that α satisfies the inequality, $\cos\alpha + t \sin\alpha \leq t$. Assume, for contradiction, that two edges $\{u, v\}$ and $\{u, w\}$ are added in Step 5, where $|uv| \leq |uw|$, and both lie in the same cone (with angle $\theta \leq \alpha$). Since the edge $\{u, v\}$ was processed first, it is already in the graph when $\{u, w\}$ is considered. For the edge $\{u, w\}$ to be prevented from insertion into H, the path $P_{u,w}$ going through v must provide a t-path in H: $|P_{u,w}| \leq t \cdot |uw|$.

We analyze the worst-case geometric configuration where v and w are located on the two distinct bounding rays of the cone, maximizing the distance $|vw|$ for angle α. Refer to Figure 3. An upper bound on $|P_{u,w}|$ can be given by the distance projected onto the ray containing v plus

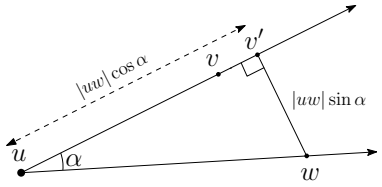


Figure 3 Illustrating the worst case path from u to w.

the stretch factor t applied to the orthogonal component: $|P_{u,w}| \leq |uw|\cos\alpha + t \cdot |uw|\sin\alpha$. To prevent the insertion of the edge $\{u, w\}$ in H, this upper bound must be less than or equal to $t \cdot |uw|$. Thus, $|uw|\cos\alpha + t \cdot |uw|\sin\alpha \leq t \cdot |uw|$. Dividing both sides by $|uw|$, we obtain the necessary angular constraint: $\cos\alpha + t\sin\alpha \leq t$. We conclude that there are at most $\lceil 2\pi/\alpha \rceil$ edges placed that are incident on u in H. For fixed t, $\lceil 2\pi/\alpha \rceil = O(1)$. Hence, Step 5 places $O(n)$ edges.

The t-spanner property of H follows directly from Steps 3, 4, 5. In Step 3, we ensure t-paths between all point pairs inside every leaf of T. In Steps 4 and 5, we ensure t-paths between all point pairs $u, v \in P$ where they belong to distinct leaves of T.

Thus, H is a t-spanner with $O(n)$ edges. ◀

Time complexity. Step 1 runs in $O(n \log n)$ time as Delaunay triangulations can be constructed in $O(n \log n)$ time (via Voronoi diagram) having at most $3n$ edges and Prim's algorithm on $O(n)$ edges runs to completion in $O(n \log n)$ time. Next, the construction of the Θ-Graph G_Θ takes $O(n \log n)$ time [28]. Also, $|E_\Theta| = O(n)$. For every pair $\{u, v\} \in E_\Theta$, verifying the existence of a t-path between u, v takes $O(n) + O(n \log n) = O(n \log n)$ time in the worst-case due to the use of Greedy-Path that runs in $O(n)$ time followed by A^* that runs in $O(n \log n)$ time in the worst-case. For $O(n)$ pairs in E_Θ, the total time taken is $O(n^2 \log n)$. So, Step 2 runs in $O(n^2 \log n)$ time. In Step 3, constructing T takes $O(n^2)$ time in the worst case. The remaining part of this step takes $O(n)$ time since the size of every leaf is at most $\ell = O(1)$. For every leaf of T, there are $O(1)$ point pairs not having t-paths. For $O(n)$ leaves, there are $O(n)$ such point pairs. Detecting such pairs takes $O(n)$ time since the size of every H_σ is $O(1)$. Finally, Steps 4 and 5 each take $O(n^3 \log n)$ time since there are $O(n^2)$ non-empty leaf pairs of T and for every such pair we spend $O(n \log n)$ time for detecting t-paths between $\leq \ell^2 = O(1)$ point pairs. Thus, the overall worst-case time complexity of Θ-Greedy is $O(n^3 \log n)$.

Remark on high time complexity. Although Θ-Greedy has a worst-case complexity of $O(n^3 \log n)$, this bound is highly pessimistic. It assumes that Steps 4 and 5 spend $O(n \log n)$ time for every leaf-pair due to A^* calls. In practice, this scenario rarely occurs. First, A^* is seldom invoked because Greedy-Path successfully finds t-paths in most cases. When Greedy-Path is used, its directional exploration visits only a small fraction of vertices before terminating, making each check extremely fast. Second, when a long t-path is discovered between two distant leaves σ_i, σ_j, it typically suffices for almost all point pairs $p \in \sigma_i, q \in \sigma_j$, greatly reducing the number of subsequent calls. Combined with bridge reuse and shortcut caching, these effects make Steps 4 and 5 far cheaper than the worst-case analysis suggests. As a result, Θ-Greedy exhibits near-quadratic scaling in practice, as confirmed by our experiments on diverse synthetic and real-world datasets in Section 3.

Space complexity. The extra space needed by Prim's algorithm and the Delaunay triangulation construction in Step 1 is $O(n)$. Constructing and storing G_Θ takes $O(n)$ space. Greedy-Path and A^* takes $O(n)$ extra space. The data structure `shortcuts` uses $O(n)$ space since, in the worst case, it can contain one t-path per edge in G_Θ. The quad tree T in Step 4 uses $O(dn)$ space, where d is its height. So, the extra space required by Θ-Greedy is $O(dn+|E'|) = O(dn)$, since $|E'| \leq n$. Thus, Θ-Greedy behaves like a linear-space algorithm in practice, making it possible to generate spanners on point sets where n is as large as 10^6.

3 Experiments

Setup. The algorithms were implemented in `C++23` using `GCC 13` and the `CGAL 6.1` library [33]. The `boost` library was used for common algorithms and data structures. Refer to Table 2. All experiments were conducted on a desktop running `Ubuntu Linux 24.04 LTS`, equipped with an Intel Core `i9-12900K` processor and `32 GB` of RAM. From CGAL, we employed the `CGAL::Exact_predicates_inexact_constructions` kernel. The entire codebase was compiled with the `O3` optimization flag to maximize performance.

Table 2 The algorithms and data structures imported from `CGAL` and `boost`.

Algorithm/Data Structure	Implementation
Graph representation	`boost::adjacency_list`
Delaunay triangulation	`CGAL::Delaunay_triangulation_2`
Prim's algorithm	`boost::prim_minimum_spanning_tree`
Dijkstra's algorithm	`boost::dijkstra_shortest_paths_no_color_map`
Priority queue	`std::priority_queue`
Quad tree	`CGAL::Quadtree`
Quad tree leaf traversal	`CGAL::Orthtrees::Leaves_traversal`
Sorting	`std::ranges::sort`

Synthetic point set distributions. We used five point set distributions to generate random point sets. These distributions were adopted from previous experimental studies [6, 9, 23, 29, 32]. They are named `uni-square`, `normal-clustered`, `grid-random`, `annulus`, and `spokes`. Refer to Figure 4 for an illustration.

The `uni-square` distribution generates points uniformly within a square region using `CGAL::Random_points_in_square_2`. It is the most common distribution used in geometry experiments. The `normal-clustered` distribution places points in randomly positioned clusters, where each cluster follows a normal distribution with a fixed mean and standard deviation. Since the clusters are separated by long distances, shortest-path calls tend to take a long time. Hence, we chose it for our experiments. The `grid-random` distribution scatters points across a non-contiguous grid within a scaled square, with integer coordinates chosen uniformly at random. This distribution is important because of the collinearity present throughout its point sets. In Figure 4, although it resembles `uni-square`, the coordinates of the points are restricted to integers. The `annulus` distribution positions points inside a ring-shaped region defined by inner and outer radii, using real-valued uniform sampling. This distribution is important because of the large hole present inside the point sets. The `spokes` distribution organizes points along five radial spokes emanating from a central point. The vast separation between the collinear spokes is useful for testing spanner algorithms. For each of the above distributions, we considered eight values of n, selected from the set $\{1K, 2K, \ldots, 128K\}$. For every value of n, we sampled five point sets.

Real-world point sets. Real-world point sets are essential because they reflect the actual spatial distributions and irregularities encountered in practical applications. Synthetic datasets, while useful for controlled benchmarking, often assume idealized conditions, such as uniform distributions, perfect clustering, or simplified geometries, that rarely occur in reality. Real-world data introduces challenges, such as uneven point density and non-uniform patterns, which may significantly influence spanner construction time and structural properties. Testing on real-world datasets ensures that algorithms are robust, scalable, and effective under realistic conditions. In our experiments, we used six real-world datasets, as described next.

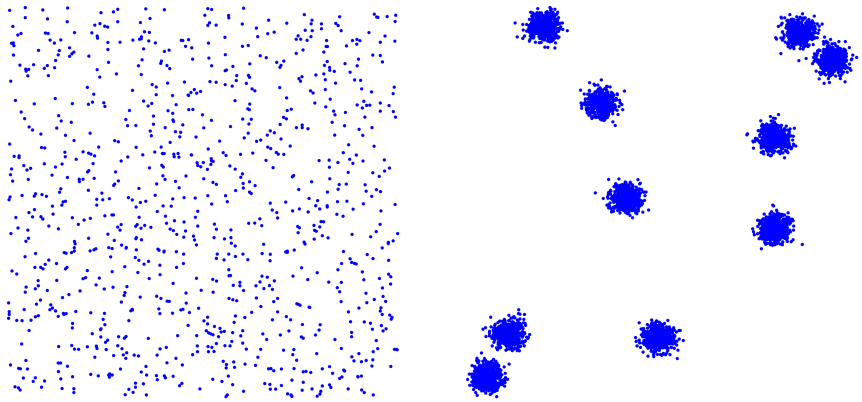
Table 3 The six real-world point sets and their sizes.

Dataset	`burma`	`birch3`	`monalisa`	`KDDCU2D`	`usa`	`europe`
n	33,708	99,999	100,000	104,297	115,475	168,435

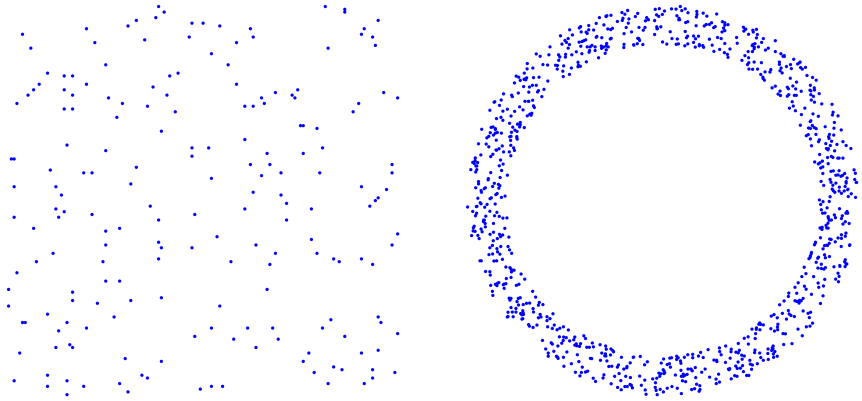
The `burma` dataset consists of 33,708 points representing cities across Burma. The `birch3` dataset contains 99,999 points arranged in random clusters scattered throughout the plane. The `monalisa` dataset is a 100,000-city TSP instance depicting a continuous-line drawing of the Mona Lisa. The `KDDCU2D` dataset includes 104,297 points corresponding to the first two dimensions of a protein dataset. The `usa` dataset represents a large-scale TSP instance with 115,475 cities covering nearly all towns, villages, and cities in the United States. Finally, the `europe` dataset comprises 168,435 points encoding differential coordinates of the European map. Refer to Table 3 for a summary.

Metrics and comparisons. Next, we compare Θ-Greedy with Bucketing using runtime, degree Δ, average degree (Δ_{avg}), and lightness as metrics.

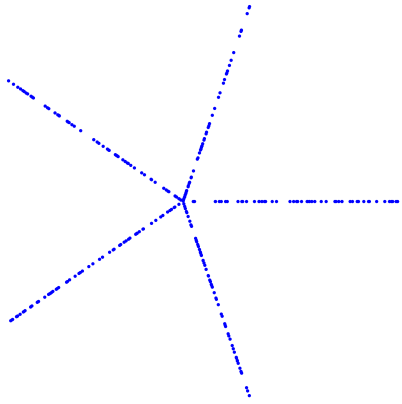
1. *Runtime.* When constructing spanners on large point sets, runtime becomes a critical concern. Guaranteeing a t-path between every pair of points remains a significant challenge. For synthetic point sets, the maximum speedup was observed when $t = 1.05$ (the smallest value used in our experiments) and $n = 128K$. Refer to Figures 5, 6, 7, 8 and 9. For every value of n, the average runtime of five runs is reported. The corresponding speedups were 62.57, 43.18, 49.16, 44.08, and 40.78 for `uni-square`, `normal-clustered`, `grid-random`, `annulus`, and `spokes` respectively. For real-world point sets (Table 4), the highest speedup of 147.59 occurred for the `europe` dataset when $t = 1.1$. As discussed next, Θ-Greedy achieved these speedups while maintaining low values of a and b, ensuring high-quality (a, b)-doppelgängers. We also observed that Bucketing could match the speed of Θ-Greedy when $t = 2$; however, in that case, Θ-Greedy can be easily accelerated further by using additional threads.
   For point sets of size 10^6, Θ-Greedy completed within a few hours (Table 6) using just one thread. Shariful et al. [32] reported that Bucketing requires about five days for the `uni-square` distribution, whereas Θ-Greedy finished in roughly two hours, corroborating its remarkable efficiency. For other distributions, Θ-Greedy consistently performed well, completing within a few hours. The slowest case was the `normal-clustered` point set, which took about four hours due to the presence of long t-paths in the final spanner, causing Greedy-Path and A^* to run slower.
2. *Degree and average degree.* Degree matters for load balancing and hardware constraints. For instance, in computer networks, a node with too many connections consumes more energy and bandwidth. Consequently, a low degree is desired. The average degree metric reflects the overall sparsity and efficiency of a network. A low average degree means fewer edges per vertex, reducing cost and complexity while maintaining connectivity.



(a) `uni-square` (b) `normal-clustered`



(c) `grid-random` (d) `annulus`



(e) `spokes`

Figure 4 The five distributions used to generate random point sets.

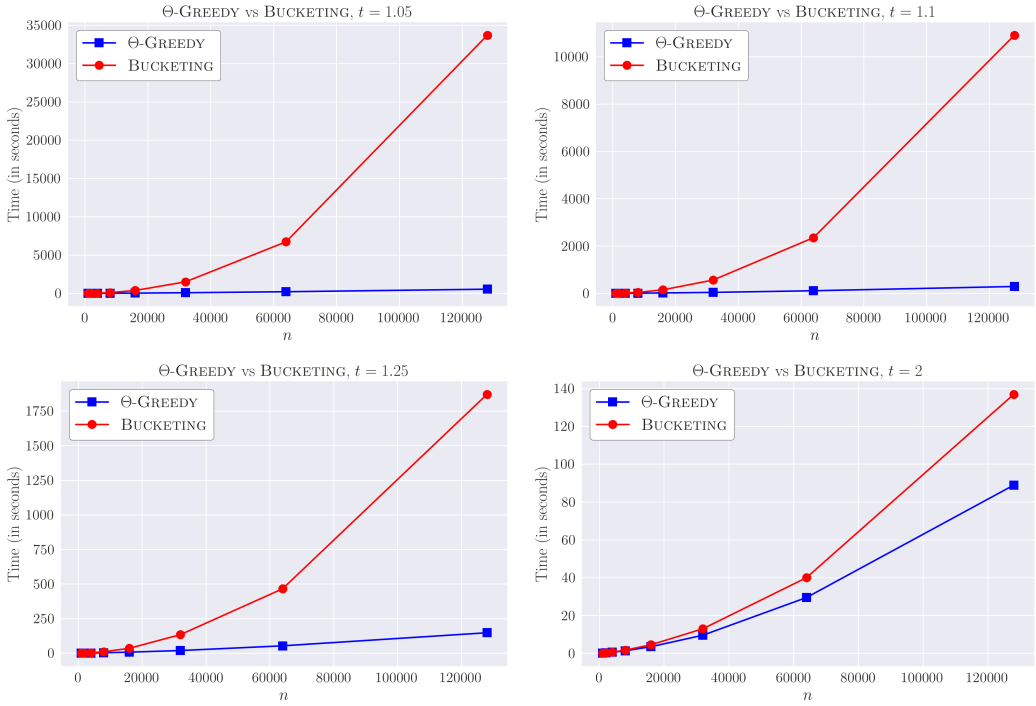


Figure 5 Θ-Greedy vs Bucketing; runtime comparison for the `uni-square` distribution.

According to the definition of (a, b)-doppelgängers, introduced in Section 1, lower values for a and b are desired. We found that for both synthetic and real-world point sets, the maximum value of a did not exceed 0.002, when all four values of t were considered; refer to Table 5. This shows that Θ-Greedy is capable of producing greedy-like sparse spanners. To our knowledge, no other practical algorithm matches this performance.
Regarding the actual difference in degrees, we observe that it is always 0 for the real-world point sets (Table 4) and at most 4 for the synthetic point sets. The maximum value of 4 occurs for the `annulus` instances, whereas for `uni-square` the difference remains 0.

3. *Lightness.* It is crucial because it controls the total cost of connectivity. In real-world networks, such as transportation systems, wireless sensor deployments, or fiber-optic layouts, lower lightness means the network uses short edges while still guaranteeing good connectivity. This translates to lower construction costs, reduced energy consumption, and easier maintenance, making spanners both efficient and practical. As with the average degree, we found that the lightness of Θ-Greedy spanners was very close to that of Bucketing spanners. Refer to Table 5. For the synthetic point sets, the maximum value of b was 0.0875, achieved for the `annulus` distribution. For the real-world point sets, it was 0.06, achieved for the `burma` dataset, when $t = 1.05$; see Table 4.

Multithreaded performance. We now describe how Θ-Greedy is parallelized using OpenMP. The construction of the Θ-Graph is naturally parallelizable, since the edges generated for each point are independent. In Step 2, we process edges from E_Θ in batches of size T, where T is the number of threads. For each batch, we run T instances of Greedy-Path in parallel. Let $T' \leq T$ denote the number of edges in the batch whose endpoints are not connected by a t-path, as determined by Greedy-Path. For these T' edges, A^* is then executed sequentially

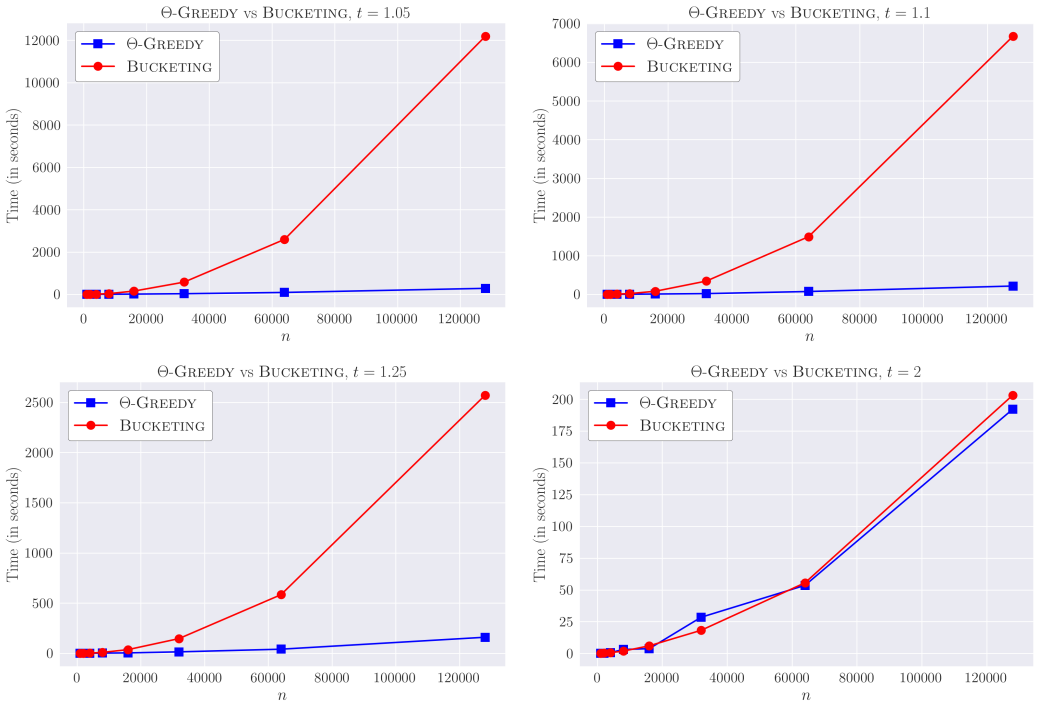


Figure 6 Θ-Greedy vs Bucketing; runtime comparison for the `normal-clustered` distribution.

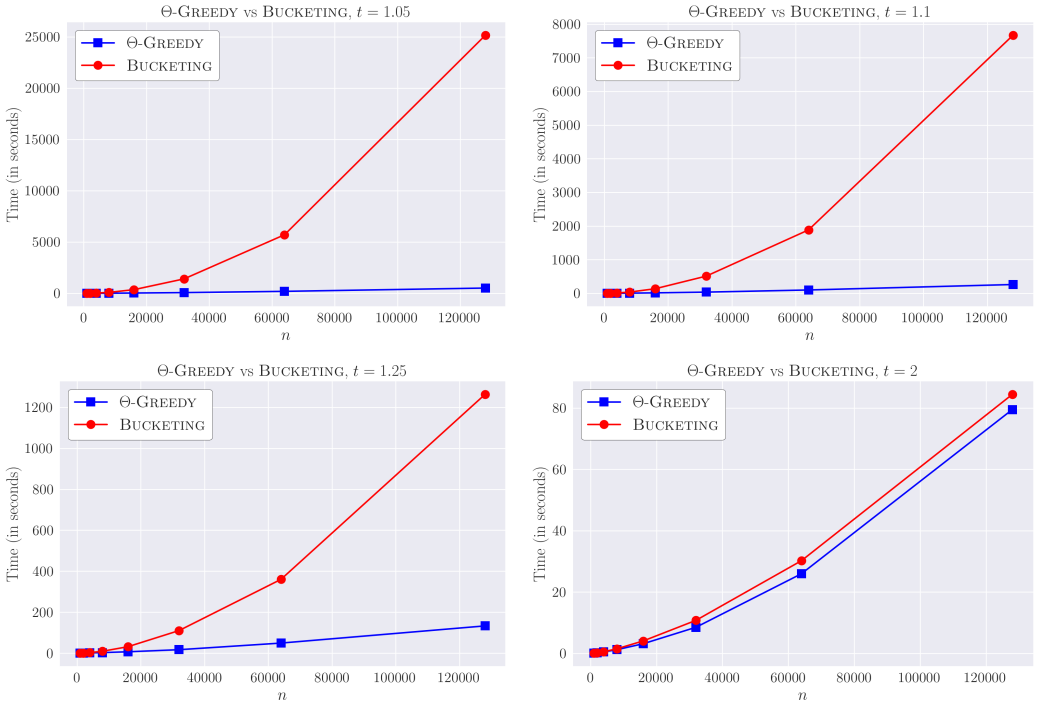


Figure 7 Θ-Greedy vs Bucketing; runtime comparison for the `grid-random` distribution.

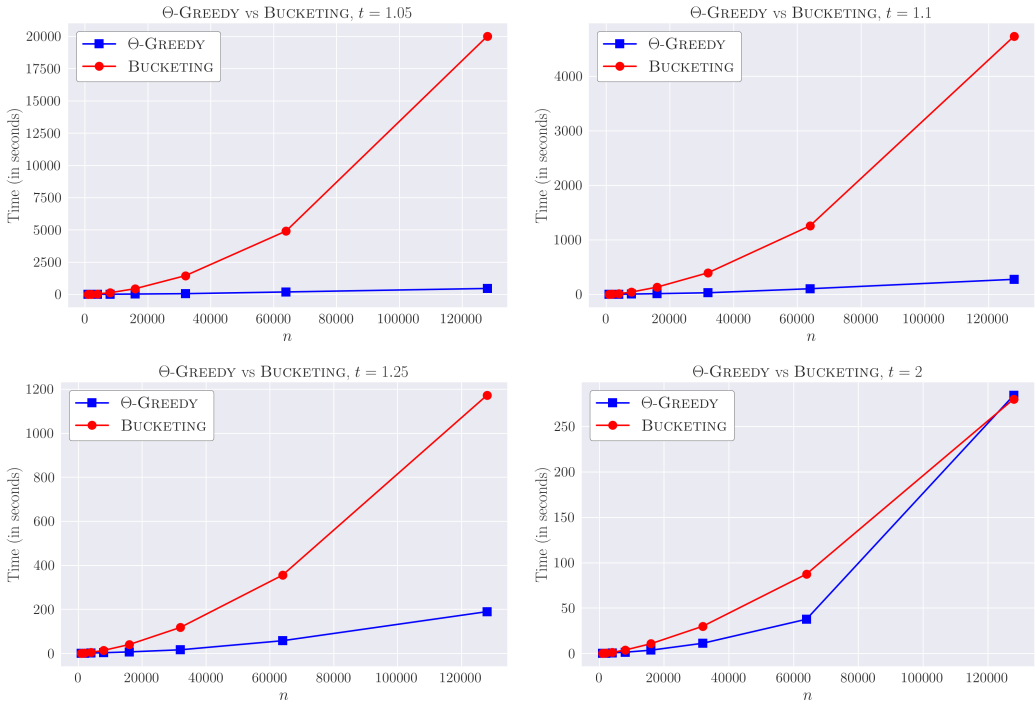


Figure 8 Θ-Greedy vs Bucketing; runtime comparison for the `annulus` distribution.

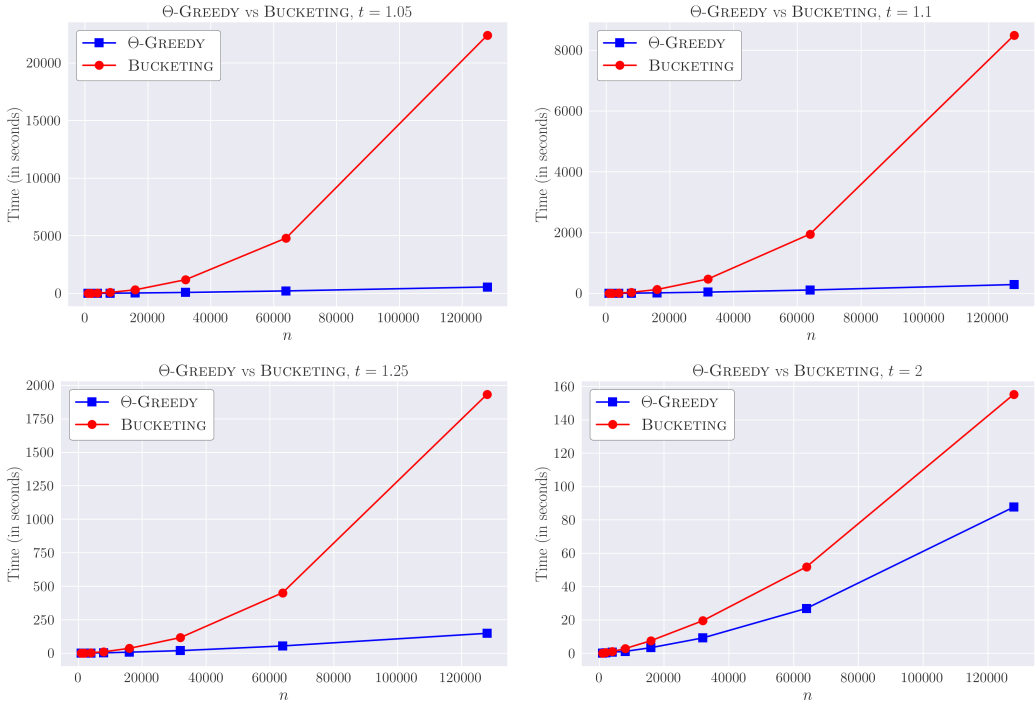


Figure 9 Θ-Greedy vs Bucketing; runtime comparison for the `spokes` distribution.

Table 4 Runtime (in minutes), degree (Δ), average degree (Δ_{avg}), and lightness comparisons are shown for the real-world point sets. $1T, 2T, 4T$ stands for $1, 2, 4$ threads, respectively.

Dataset	t	Θ-Greedy						Bucketing			
		$1T$	$2T$	$4T$	Δ	Δ_{avg}	Lightness	$1T$	Δ	Δ_{avg}	Lightness
burma	1.05	1.33	0.97	0.71	31	13.51	20.78	6.50	31	13.51	20.72
burma	1.1	0.71	0.53	0.40	19	8.97	11.45	3.23	19	8.97	11.40
burma	1.25	0.33	0.25	0.18	12	5.82	5.37	1.31	12	5.82	5.37
burma	2	0.17	0.13	0.09	5	2.93	1.98	0.45	5	2.93	1.98
birch3	1.05	4.43	3.39	2.56	27	13.04	19.91	127.97	27	13.04	19.90
birch3	1.1	2.52	1.86	1.40	18	9.04	11.36	67.41	18	9.04	11.36
birch3	1.25	1.47	1.03	0.76	11	5.49	5.38	30.23	11	5.49	5.38
birch3	2	1.17	0.78	0.55	6	2.87	2.01	9.96	6	2.87	2.00
monalisa	1.05	6.02	4.44	3.40	17	12.09	9.98	124.64	17	12.09	9.98
monalisa	1.1	3.02	2.21	1.69	13	8.92	6.24	38.33	13	8.92	6.24
monalisa	1.25	1.46	1.04	0.77	9	6.37	3.79	5.45	9	6.37	3.79
monalisa	2	0.85	0.56	0.39	6	3.65	1.98	0.83	6	3.65	1.98
KDDCU2D	1.05	5.97	4.54	3.36	27	13.31	20.27	66.20	27	13.31	20.27
KDDCU2D	1.1	3.08	2.32	1.72	20	9.14	11.59	33.75	20	9.14	11.56
KDDCU2D	1.25	1.57	1.13	0.83	11	5.49	5.39	14.36	11	5.49	5.39
KDDCU2D	2	0.97	0.62	0.43	6	2.87	1.99	5.08	6	2.87	1.99
usa	1.05	7.53	5.80	4.44	34	13.19	19.83	51.75	34	13.19	19.78
usa	1.1	3.91	2.95	2.25	22	9.29	11.32	25.69	22	9.29	11.31
usa	1.25	2.00	1.43	1.07	12	5.76	5.35	9.90	12	5.76	5.34
usa	2	1.28	0.82	0.55	5	2.97	1.98	2.41	5	2.97	1.98
europe	1.05	14.49	10.04	7.32	30	13.10	18.17	1825.97	30	13.10	18.17
europe	1.1	7.24	5.07	3.72	19	9.03	10.59	1068.55	19	9.03	10.59
europe	1.25	3.68	2.49	1.80	12	5.47	5.10	540.75	12	5.47	5.10
europe	2	2.36	1.54	1.03	6	2.86	1.95	291.93	6	2.86	1.95

Table 5 The maximum observed values for a and b with respect to the spanners produced by Bucketing.

	uni-square	normal-clustered	grid-random	annulus	spokes	real-world
max(a)	0.0000	0.0000	0.0020	0.0010	0.0005	0.0000
max(b)	0.0000	0.0111	0.0056	0.0875	0.0030	0.0600

to find t-paths and insert edges in H whenever required. Since Greedy-Path successfully finds t-paths in most cases where such paths exist in H, calls to A^* are comparatively infrequent. Detection of violating point pairs (i.e., pairs lacking t-paths) in the leaf nodes is independent in Step 3 and therefore parallelizable. In Steps 4 and 5, all leaf pairs (σ, σ_i), where σ is fixed, are also processed in parallel.

The above approach yielded speedups proportional to T, making Θ-Greedy scalable for large point sets. Refer to Figures 10, 11, 12, 13, 14 and Tables 4 and 6. We observed consistent speedups across all values of t used in our experiments.

Comparison with Fast-Sparse-Spanner. Since Fast-Sparse-Spanner [32] was the only previously known practical algorithm for constructing spanners on large point sets, it is natural to compare our algorithm against it. Since one run of Bucketing on a million points takes several days (roughly 5 days for `uni-square`), it is impractical for comparison purposes. However, the trends show that Θ-Greedy spanners are still competitive with Bucketing spanners even for massive point sets. For instance, for the `uni-square` point set, we found the average degree to be 9.1, which is roughly the same as that of the $100K$-element point set used in Section 1 (the reported value for Bucketing is 9.08).

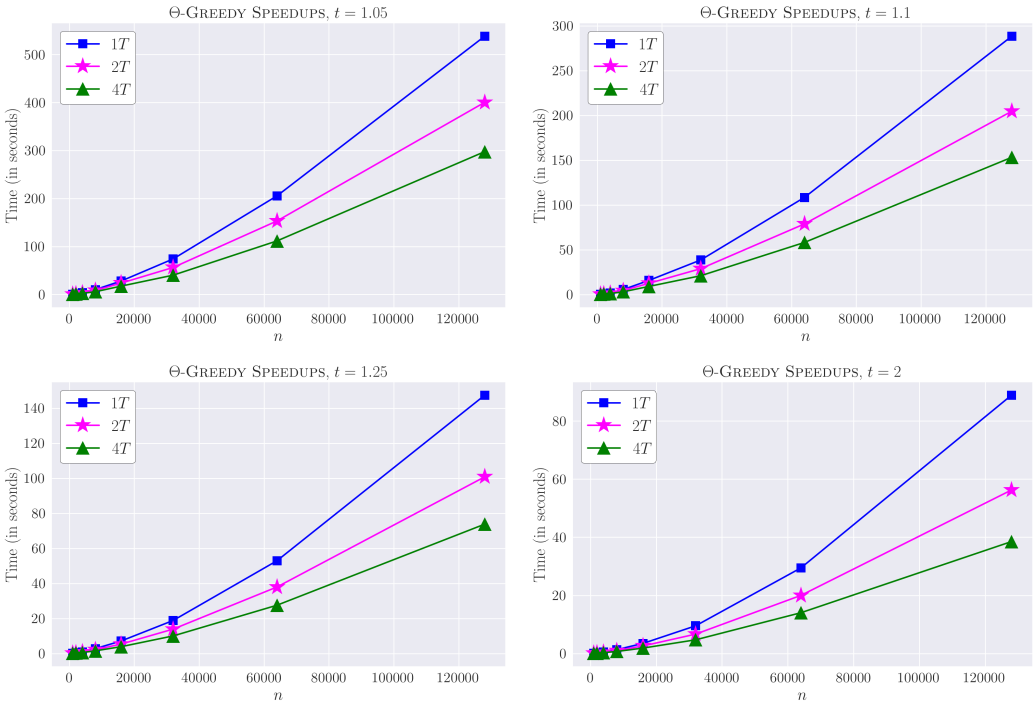


Figure 10 Θ-Greedy: multithreaded runtimes for the `uni-square` distribution.

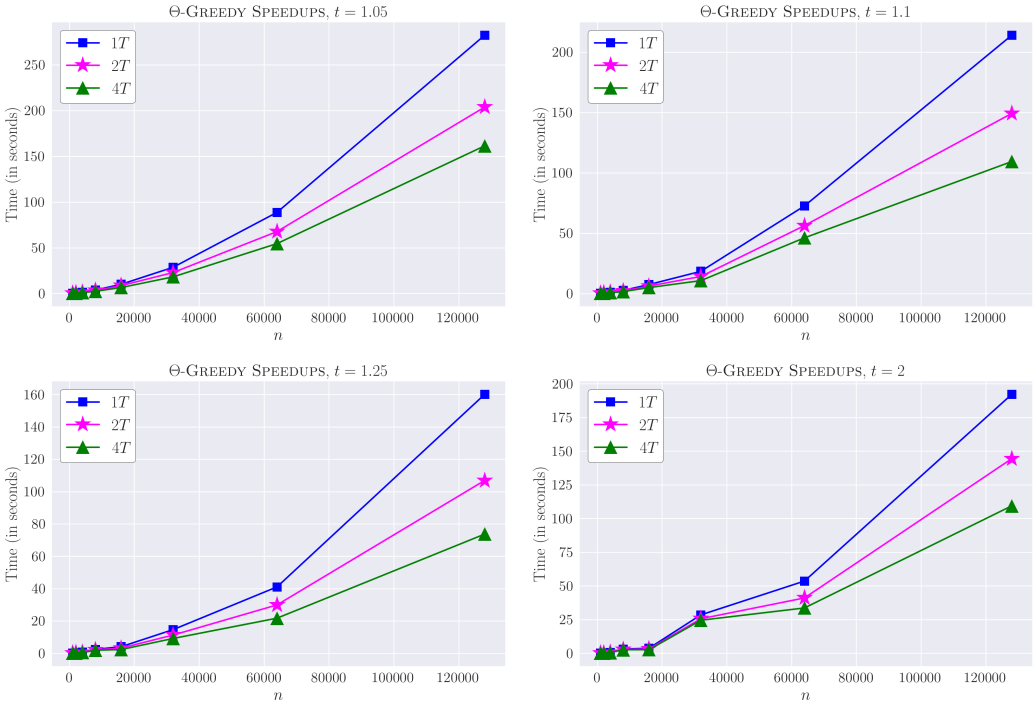


Figure 11 Θ-Greedy: multithreaded runtimes for the `normal-clustered` distribution.

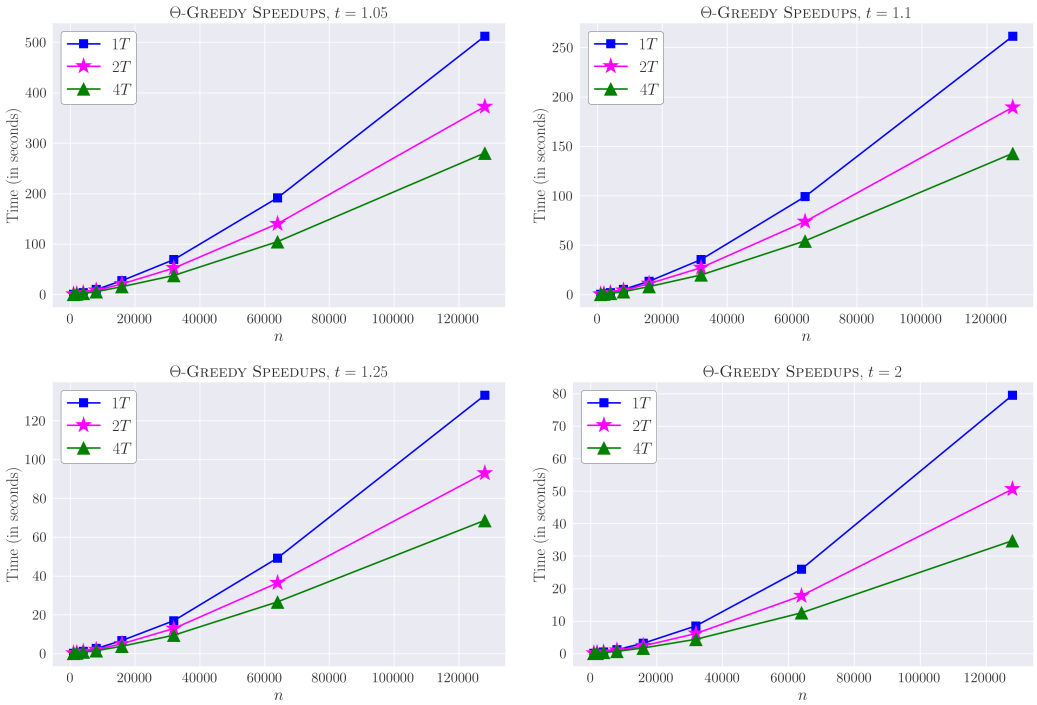


Figure 12 Θ-Greedy: multithreaded runtimes for the `grid-random` distribution.

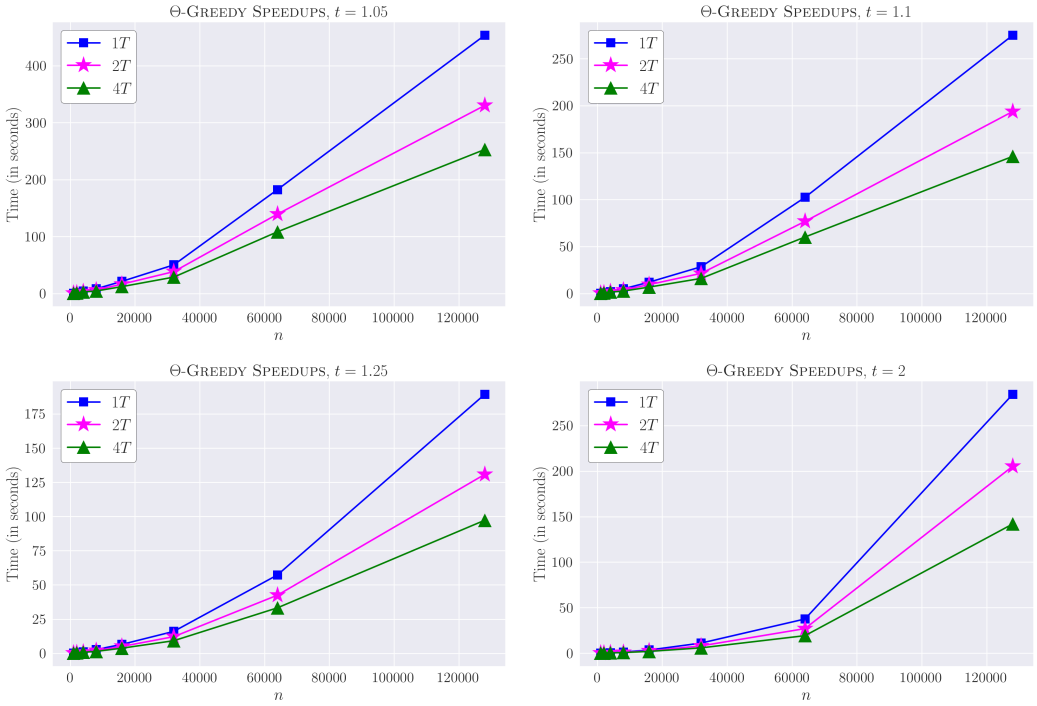


Figure 13 Θ-Greedy: multithreaded runtimes for the `annulus` distribution.

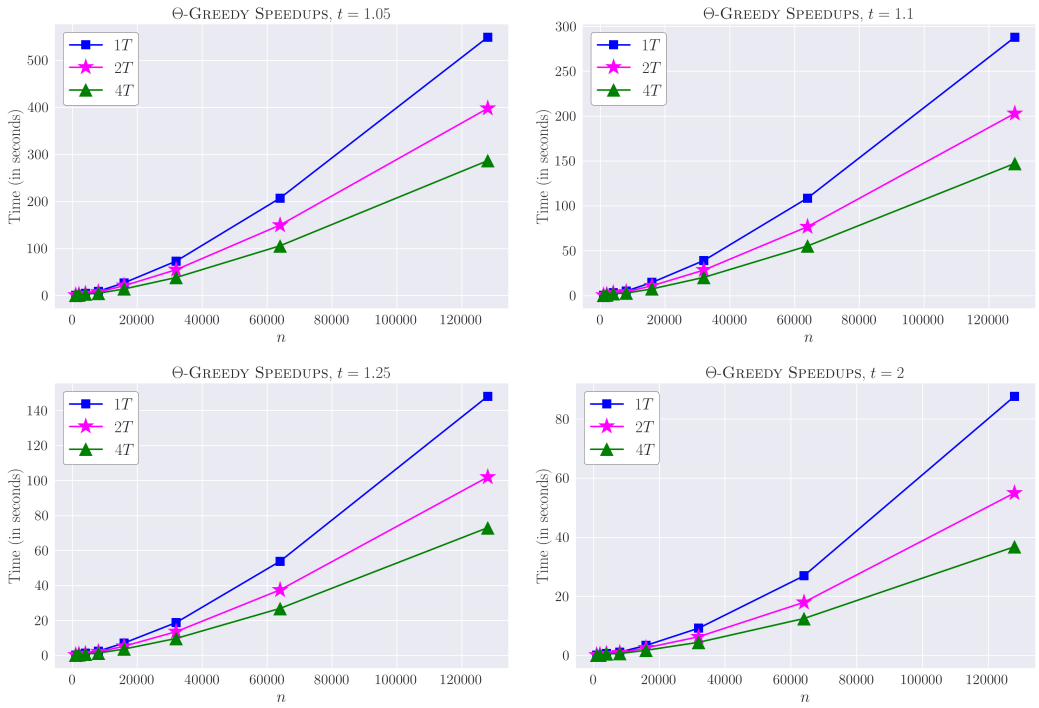


Figure 14 Θ-Greedy: multithreaded runtimes for the `spokes` distribution.

See Table 6, where we set $n = 10^6$ and $t = 1.1$. The spanners produced by Θ-Greedy have significantly lower degrees than those generated by Fast-Sparse-Spanner, indicating that the resulting spanners differ substantially. The biggest differences were achieved for the `uni-square` and `grid-random` point sets. Interestingly, for the $128K$-element `uni-square` point sets, the maximum degree was at most 19 for $t = 1.1$; the same value was observed even when $n = 10^6$. This demonstrates that our algorithm preserves greedy-like degree behavior even for massive point sets.

Although the differences in average degree are modest, the number of edges inserted by Fast-Sparse-Spanner is noticeably higher. For example, on the `uni-square` point set, Fast-Sparse-Spanner added roughly 455,000 more edges than Θ-Greedy. Moreover, the Fast-Sparse-Spanner spanners were substantially heavier than the Θ-Greedy spanners. This is primarily due to the long WSPD-spanner edges that Fast-Sparse-Spanner employs as a backbone. Taken together, these observations clearly demonstrate the superiority of Θ-Greedy over Fast-Sparse-Spanner.

Finally, the lightness of the Θ-Greedy spanners were significantly lower. See Table 6. The largest disparity occurred for the `normal-clustered` point set, where the lightness of the Fast-Sparse-Spanner graph was approximately 320 times greater than that of the Θ-Greedy spanner.

We observe that Fast-Sparse-Spanner is faster than ours in practice. In particular, for `uni-square`, `normal-clustered`, `grid-random`, `annulus`, and `spokes`, the time taken by the algorithm was roughly 55, 18, 57, 53, and 19.07 minutes, respectively, on a system comparable to ours using one thread. Fast-Sparse-Spanner, ran fastest on the `normal-clustered` point set due to the long WSPD-edges connecting distant clusters, owing to which t-path verifications terminated much faster compared to ours. In our case, the opposite happened,

Table 6 Θ-Greedy vs Fast-Sparse-Spanner [32]: runtime (in minutes), degree (Δ) and average degree (Δ_{avg}), when $n = 10^6$ and $t = 1.1$. $1T, 2T, 4T$ stands for $1, 2, 4$ threads, respectively.

$n = 10^6$, $t = 1.1$	uni-square	normal-clustered	grid-random	annulus	spokes
Θ-Greedy $1T$ Runtime	123.69	239.32	124.44	188.76	125.86
Θ-Greedy $2T$ Runtime	89.57	160.6	88.22	125.61	86.73
Θ-Greedy $4T$ Runtime	69.31	108.47	70.51	88.3	65.02
Θ-Greedy Δ	19	19	19	19	18
Fast-Sparse-Spanner Δ	1002	625	1003	573	972
Θ-Greedy Δ_{avg}	9.1	8.84	9.1	9.07	9.1
Fast-Sparse-Spanner Δ_{avg}	9.97	9.15	9.84	9.6	9.88
Θ-Greedy Lightness	11.08	9.88	11.07	11.67	11.07
Fast-Sparse-Spanner Lightness	238.62	3164.69	236.36	164.9	213.22

and we observed that Θ-Greedy was slowest on the same point set and took almost 4 hours to finish due to the presence of much longer t-paths between distant clusters. With the increase in the number of threads (T), Θ-Greedy could accelerate in all five cases.

4 Conclusions

We introduced Θ-Greedy, a practical parallel algorithm for constructing doppelgängers of greedy geometric spanners that are structurally and metrically almost indistinguishable from the actual greedy spanners. Thus, Θ-Greedy spanners serve as practical and reliable alternatives to true greedy spanners. In our experiments, we found that Θ-Greedy consistently achieved almost-greedy sparsity, lightness, and degree distribution while scaling to million-point instances with linear space and significant speedups over Bucketing, the fastest known greedy spanner algorithm. Our results close a long-standing gap between theoretical optimality and practical feasibility for low stretch factors, offering the first engineered solution that combines simplicity, efficiency, accuracy, and scalability for large real-world geometric networks.

An exciting avenue for future work is reducing the worst-case time complexity of Θ-Greedy without sacrificing its ability to produce doppelgängers. Exploring distribution-sensitive optimizations and GPU acceleration could further broaden its applicability. Additionally, extending these ideas to higher dimensions remains a compelling open problem.

References

1 A Karim Abu-Affash, Gali Bar-On, and Paz Carmi. δ-greedy t-spanner. *Computational Geometry*, 100:101807, 2022. doi:10.1016/J.COMGEO.2021.101807.

2 Sander Alewijnse, Quirijn W Bouts, Alex P ten Brink, and Kevin Buchin. Computing the greedy spanner in linear space. *Algorithmica*, 73(3):589–606, 2015. doi:10.1007/S00453-015-0001-2.

3 Sander Alewijnse, Quirijn W Bouts, Alex P Ten Brink, and Kevin Buchin. Distribution-sensitive construction of the greedy spanner. *Algorithmica*, 78(1):209–231, 2017. doi:10.1007/S00453-016-0160-9.

4 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993. doi:10.1007/BF02189308.

5 Khaled Alzoubi, X-Y Li, Yu Wang, P-J Wan, and Ophir Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on parallel and Distributed Systems*, 14(4):408–421, 2003. doi:10.1109/TPDS.2003.1195412.

6 Frederick Anderson, Anirban Ghosh, Matthew Graham, Lucas Mougeot, and David Wisnosky. Bounded-degree plane geometric spanners in practice. *ACM Journal of Experimental Algorithmics*, 28:1–36, 2023. doi:10.1145/3582497.

7 Sunil Arya, David M. Mount, and Michiel Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 703–712, 1994. doi:10.1109/SFCS.1994.365722.

8 Sunil Arya, David M Mount, and Michiel Smid. Dynamic algorithms for geometric spanners of small diameter: Randomized solutions. *Computational Geometry*, 13(2):91–107, 1999. doi:10.1016/S0925-7721(99)00014-0.

9 Jon Jouis Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on computing*, 4(4):387–411, 1992. doi:10.1287/IJOC.4.4.387.

10 Glencora Borradaile, Hung Le, and Christian Wulff-Nilsen. Greedy spanners are optimal in doubling metrics. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2371–2379. SIAM, 2019. doi:10.1137/1.9781611975482.145.

11 Prosenjit Bose, Paz Carmi, Mohammad Farshi, Anil Maheshwari, and Michiel Smid. Computing the greedy spanner in near-quadratic time. *Algorithmica*, 58(3):711–729, 2010. doi:10.1007/S00453-009-9293-4.

12 Prosenjit Bose, Joachim Gudmundsson, and Michiel Smid. Constructing plane spanners of bounded degree and low weight. *Algorithmica*, 42(3):249–264, 2005. doi:10.1007/S00453-005-1168-8.

13 Prosenjit Bose and Michiel Smid. On plane geometric spanners: A survey and open problems. *Computational Geometry*, 46(7):818–830, 2013. doi:10.1016/J.COMGEO.2013.04.002.

14 Paul B Callahan and S Rao Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *SODA*, volume 93, pages 291–300, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313777.

15 Paul B Callahan and S Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM (JACM)*, 42(1):67–90, 1995. doi:10.1145/200836.200853.

16 Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New sparseness results on graph spanners. In *Proceedings of the eighth annual symposium on Computational geometry*, pages 192–201, 1992. doi:10.1145/142675.142717.

17 Ken Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 56–65, 1987.

18 Artur Czumaj and Hairong Zhao. Fault-tolerant geometric spanners. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 1–10, 2003. doi:10.1145/777792.777794.

19 Gautam Das and Giri Narasimhan. A fast algorithm for constructing sparse euclidean spanners. *International Journal of Computational Geometry & Applications*, 7(04):297–315, 1997. doi:10.1142/S0218195997000193.

20 Adrian Dumitrescu and Anirban Ghosh. Lower bounds on the dilation of plane spanners. *International Journal of Computational Geometry & Applications*, 26(02):89–110, 2016. doi:10.1142/S0218195916500059.

21 William Evans and Lucca Morais de Arruda Siaudzionis. On path-greedy geometric spanners. *Computational Geometry*, 110:101948, 2023. doi:10.1016/J.COMGEO.2022.101948.

22 Mohammad Farshi and Joachim Gudmundsson. Experimental study of geometric t-spanners. *Journal of Experimental Algorithmics (JEA)*, 14:1–3, 2010.

23 Rachel Friederich, Anirban Ghosh, Matthew Graham, Brian Hicks, and Ronald Shevchenko. Experiments with unit disk cover algorithms for covering massive pointsets. *Computational Geometry*, 109:101925, 2023. doi:10.1016/J.COMGEO.2022.101925.

24 Jie Gao, Leonidas J Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanner for routing in mobile networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 45–55, 2001. doi:10.1145/501416.501424.

25 Anirban Ghosh, FNU Shariful, and David Wisnosky. Visualizing WSPDs and Their Applications. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:4, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2022.68.

26 Hung Le and Cuong Than. Greedy spanners in euclidean spaces admit sublinear separators. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3287–3310. SIAM, 2022. doi:10.1137/1.9781611977073.130.

27 James D Marble and Kostas E Bekris. Asymptotically near-optimal planning with probabilistic roadmap spanners. *IEEE Transactions on Robotics*, 29(2):432–444, 2013. doi:10.1109/TRO.2012.2234312.

28 Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

29 Giri Narasimhan and Martin Zachariasen. Geometric minimum spanning trees via well-separated pair decompositions. *Journal of Experimental Algorithmics (JEA)*, 6:6–es, 2001.

30 Satish B Rao and Warren D Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 540–550, 1998.

31 Daniel Russel and L Guibas. Exploring protein folding trajectories using geometric spanners. In *Biocomputing 2005*, pages 40–51. World Scientific, 2005.

32 FNU Shariful, Justin Weathers, Anirban Ghosh, and Giri Narasimhan. Engineering an algorithm for constructing low-stretch geometric graphs with near-greedy average degrees. *Computational Geometry*, 130:102201, 2026. doi:10.1016/J.COMGEO.2025.102201.

33 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 6.1 edition, 2025. URL: https://doc.cgal.org/6.1/Manual/packages.html.

34 Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. CRC press, 2017.

35 Ge Xia. The stretch factor of the delaunay triangulation is less than 1.998. *SIAM Journal on Computing*, 42(4):1620–1659, 2013. doi:10.1137/110832458.

Approximating Pareto Sum via Bounded Monotone Min-Plus Convolution

Geri Gokaj
Karlsruhe Institute of Technology, Germany

Marvin Künnemann
Karlsruhe Institute of Technology, Germany

Sabine Storandt
University of Konstanz, Germany

Carina Truschel
University of Konstanz, Germany

Abstract

The Pareto sum of two-dimensional point sets P and Q in $\mathbb{R}^2$ is defined as the skyline of the points in their Minkowski sum. The problem of efficiently computing the Pareto sum arises frequently in bi-criteria optimization algorithms. Prior work establishes that computing the Pareto sum of sets P and Q of size n suffers from conditional lower bounds that rule out strongly subquadratic $O(n^{2-\epsilon})$-time algorithms, even when the output size is $\Theta(n)$. Naturally, we ask: How efficiently can we *approximate* Pareto sums, both in theory and practice? Can we beat the near-quadratic-time state of the art for exact algorithms?

On the theoretical side, we formulate a notion of additively approximate Pareto sets and show that computing an approximate Pareto set is *fine-grained equivalent* to Bounded Monotone Min-Plus Convolution. Leveraging a remarkable $\tilde{O}(n^{1.5})$-time algorithm for the latter problem (Chi, Duan, Xie, Zhang; STOC '22), we thus obtain a strongly subquadratic (and conditionally optimal) approximation algorithm for computing Pareto sums.

On the practical side, we engineer different algorithmic approaches for approximating Pareto sets on realistic instances. Our implementations enable a granular trade-off between approximation quality and running time/output size compared to the state of the art for exact algorithms established in (Funke, Hespe, Sanders, Storandt, Truschel; Algorithmica '25). Perhaps surprisingly, the (theoretical) connection to Bounded Monotone Min-Plus Convolution remains beneficial even for our implementations: in particular, we implement a simplified, yet still subquadratic version of an algorithm due to Chi, Duan, Xie and Zhang, which on some sufficiently large instances outperforms the competing quadratic-time approaches.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Computational geometry

Keywords and phrases computational geometry, fine-grained complexity, algorithm engineering

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.54

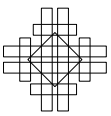
Related Version *Full Version*: `https://arxiv.org/pdf/2603.25449`

Supplementary Material *Software (Source Code)*: `https://github.com/ggokaj/SoCG26-ParetoSum` archived at `swh:1:dir:53715b7d04ec81c190b784371b4675bc28183aee`
Dataset (Benchmark): `https://doi.org/10.5281/zenodo.19260269`

Funding *Geri Gokaj*: Partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 462679611.
Marvin Künnemann: Partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 462679611.
Carina Truschel: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 251654672 – TRR 161.



© Geri Gokaj, Marvin Künnemann, Sabine Storandt, and Carina Truschel;
licensed under Creative Commons License CC-BY 4.0



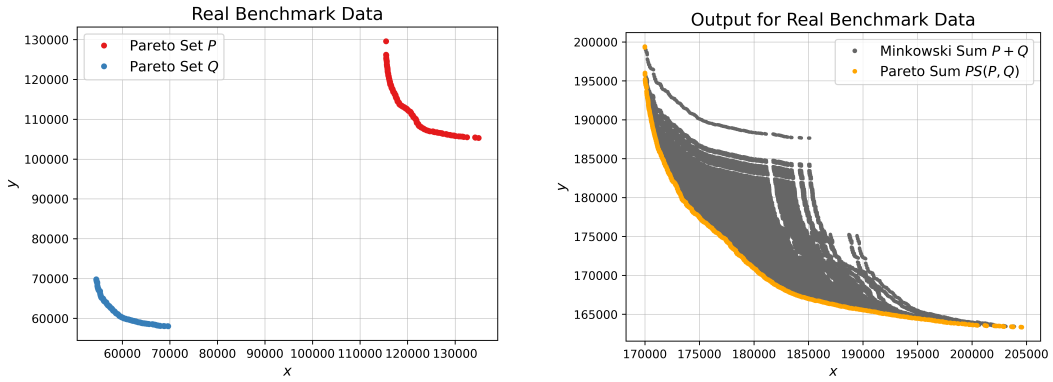


Figure 1 Two Pareto sets P, Q on the left arising from bi-criteria route planning. Depicted on the right is the Minkowski sum $P + Q$ with the Pareto sum $PS(P, Q)$.

1 Introduction

In multi-criteria decision-making processes and in algorithms for multi-objective optimization, a central computational task is to obtain and filter non-dominated solutions. This is essential in order to keep intermediate solution sets interpretable and computationally tractable. Often, partial solutions must be combined in the course of the algorithm, and non-dominated filtering must be applied to the resulting set. This core problem is known as *Pareto sum*: Given two sets $P, Q \subset \mathbb{R}^2$ of non-dominated points[1] (also called skylines or Pareto sets), the Pareto sum $PS(P, Q)$ is defined as the skyline of their Minkowski sum $P + Q := \{p + q \mid \ p \in P, q \in Q\}$.

Recently, exact algorithms have been developed for Pareto sum computation with a running time in $\tilde{O}(n^2)$ [13]. Naturally, the question arises whether this algorithm admits the best-possible time complexity for this problem or whether there is hope for designing asymptotically faster ones. The study of fine-grained complexity has emerged as a powerful framework for understanding the algorithmic complexity of problems within the class P. Using fine-grained reductions, the conjectured intractability of one key problem can be transferred to another problem, yielding a conditional lower bound on how fast the other problem can be solved. For the Pareto sum problem, it was recently shown that an algorithm with a running time in $\mathcal{O}(\min\{n^2, nk\}^{1-\delta})$ for $\delta > 0$, where k denotes the output size, would disprove the 3-SUM hypothesis [15]. The conditional lower bound matches the complexity of known algorithms (up to subpolynomial factors), which renders the existence of subquadratic algorithms for Pareto sum (even with linear output sizes) unlikely.

However, in practical applications, Pareto sum instances of substantial input size n and with output sizes $k \in \Omega(n)$ frequently arise. On such instances, exact computation takes at least quadratic time, which is typically impractical. This motivates our study of approximation algorithms for Pareto sum. We define a point set $\tilde{S}$ to be an approximation of $S := PS(P, Q)$ if for each element s in S there is a point $\tilde{s}$ in $\tilde{S}$ of "similar quality" (specifically, $\tilde{s} \leq s+(\Delta, \Delta)$) and each $\tilde{s} \in \tilde{S}$ is indeed close to a point in $P + Q$ (specifically $p + q \leq \tilde{s} \leq p + q + (\Delta, \Delta)$ for some $p \in P, q \in Q$). This provides an additive approximation guarantee. This definition caters well to many practical applications of Pareto sum computation. For example, in bi-criteria route planning, the goal is to find the set of Pareto-optimal paths between the source and the target node with respect to two metrics. Route planning techniques for this

[1] We call a set S of points non-dominated, if there do not exist distinct vectors $s, s' \in S$ such that $s \leq s'$.

setup heavily rely on Pareto sum computations and the number of resulting skyline points is oftentimes huge [13, 16, 25]. Thus, the goal is to reduce the number of Pareto-optimal paths to be considered within the algorithm and also as an output for the user. Multiplicative approximation algorithms tend to not work well in this scenario, because for paths with a value of zero or close to zero in one metric, even slightly suboptimal paths induce huge approximation factors. Additive approximations do not suffer from the same issue and are also much easier to interpret for a user (for example, if it is guaranteed that the reported route has at most 2 minutes of extra driving time). In this paper, we propose an efficient additive-approximation algorithm that allows one to trade running time against solution quality. We rigorously analyze its theoretical properties and also provide an extensive empirical study of its performance on synthetic and real data sets.

1.1 Related Work

The Pareto sum problem has already been extensively studied from a theoretical and practical perspective. A simple approach follows by computing the Minkowski sum explicitly and then applying general algorithms for skyline computation [17, 10]. In [18], the practical relevance of the problem was emphasized, and novel algorithms were proposed that achieve a subquadratic space consumption for certain input set structures. Algorithms with guaranteed output-sensitive space consumption were proposed in [13]. A simple Sort & Compare algorithm was proven to run in $\mathcal{O}(n^2 \log n)$ on input sets of size n, and a sweep search algorithm to run in $\mathcal{O}(nk + n \log n)$ where k denotes the output size. Experiments on synthetic instances and instances stemming from bi-criteria route planning in road networks confirm that these algorithms outperform those that fully compute the Minkowski sum and also the ones proposed in [18]. In [19], Pareto sum computation is used as a subroutine for multiobjective spanning tree or TSP computation which uses a tree decomposition as basis. For join nodes, an algorithm similar to Sort & Compare is used to obtain the desired combined result. In [14], several additional fine-grained complexity results for Pareto sum are provided. This includes a quadratic lower bound for Pareto sum computation even for $k = \Theta(n)$ under the 3-SUM hypothesis (which is even more believable than the previous min-plus convolution quadratic lower bound by [13], as the min-plus convolution hypothesis implies the 3-SUM hypothesis). Moreover, they established Pareto sum as a complete problem for a wide class of logical and geometric problems, further adding interest to the problem from the side of complexity theory.

1.2 Our Contributions

In this paper, we explore the degree to which the quadratic-time bottleneck for the Pareto sum problem – which is observable both in practice [13] and substantiated by conditional lower bounds [13, 14, 15] – can be circumvented by additively *approximating* Pareto sums. To this end, in Section 3, we formally introduce a natural relaxation of Pareto sums to Δ-approximate Pareto sums: intuitively, $\tilde{S}$ is an approximation of the true Pareto set S of P, Q if for any $s \in S$, we have an approximate point $\tilde{s}$ of "similar quality" (specifically, $\tilde{s} \leq s + (\Delta, \Delta)$). Conversely, any such $\tilde{s} \in \tilde{S}$ is of similar quality to some actual point in $P + Q$ (i.e., there are $p \in P, q \in Q$ with $p + q \leq \tilde{s} \leq p + q + (\Delta, \Delta)$).[2]

[2] In fact, we often even achieve the stronger guarantee that $\tilde{S} \subseteq P + Q$, see Definition 5.

Crucially, our results rely on establishing a certain fine-grained *equivalence* between approximating Pareto sums and Bounded Monotone Min-Plus Convolution, the latter of which has received increasing interest in recent years [9, 11, 4, 6, 12]. This leads to the following theoretical results:

- For any sets P, Q of at most n points in $[0, W]^2$, we can compute an ϵW-approximate Pareto sum $\tilde{S}$ in time $\tilde{O}(n + (1/\epsilon)^{1.5})$. This leverages a remarkable algorithm due to Chi, Duan, Xie and Zhang [11] solving Bounded Monotone Min-Plus Convolution in strongly subquadratic time $\tilde{O}(n^{1.5})$.
- Conversely, we show that any $\tilde{O}(n + (1/\epsilon)^{1.5-\delta})$-time algorithm for computing an ϵW-approximate Pareto sum would give a polynomial-factor improvement for Bounded Monotone Min-Plus Convolution. This establishes a fine-grained equivalence of the approximate Pareto sum and Bounded Min-Plus Convolution. Some approximation problems [8, 7] are known to be fine-grained equivalent to min-plus convolution, but this appears to be the first such connection for bounded monotone min-plus convolution.
- We observe that slightly stronger approximation guarantees can be obtained if a *witness-reporting* version of Bounded Monotone Min-Plus Convolution can be achieved. Indeed, we show that such witnesses can be reported in strongly subquadratic time, by adapting a $\tilde{O}(n^{1.6})$-time algorithm for Bounded Monotone Min-Plus Convolution [11] using techniques due to Alon and Naor [1].

Subsequently, we observe how beneficial our relaxation to approximate Pareto sums can be on realistic input sets. To this end, we empirically fine-tune and evaluate a number of different approaches, leading to the following results:

- On synthetic and realistic inputs, we can approximate Pareto sums significantly more efficiently than computing them exactly. E.g., on generated data sets with 10^6 points and coordinates in $[0, 2 \cdot 10^6]$, computing an additive ($\Delta = 20$)-approximation saves a computation time of up to three orders of magnitude. Furthermore, the size of the returned sets reduces significantly with coarser approximation parameters. Thus, we enable an effective choice of running time/solution size vs. approximation quality.
- We focus on efficiently solving the *Bounded* Pareto sum problem, which denotes the special case in which all coordinates are integer values in $\{0, \dots, W-1\}$ – such instances occur naturally in our approximation algorithms (after a scaling step), but are of independent interest in bi-criteria optimizations with integral cost functions (e.g., in routing). Here, we empirically evaluate and compare several algorithmic approaches: (1) a natural adaptation of the Sort & Compare method of [13] to BucketSort & Compare, (2) a convex-pruning algorithm inspired by a fast algorithm for Near-convex Min-Plus Convolution [5], and (3) an algorithm based on a $\tilde{O}(n^{1.6})$-time algorithm for Bounded Monotone Min-Plus Convolution [11] (which we will refer to as CDXZ algorithm). Perhaps surprisingly, we observe that depending on the class of inputs, either one of these approaches may outperform the others. This indicates that the (theoretical) connection between Pareto sums and Bounded Monotone Min-Plus convolution can also be exploited in practice to obtain practical algorithms (depending on the instances).

We feel that our results add practical evidence (and extend theoretical evidence, e.g. [21, 8, 3]) that studying the connections of geometric (and other) problems with certain formulations of convolution-type problems is beneficial for algorithmic research.

2 Bounded Pareto Sum

In this section, we prove our core reduction which is the basis for developing new algorithms for (approximate) Pareto sum computation. Throughout the paper, we abbreviate $[n] := \{0, 1, \ldots, n-1, n\}$ and use the interval notation $[a, b] := \{a, a+1, \ldots, b-1, b\}$ for some $a, b \in \mathbb{Z}$, with $a \leq b$. We say that a point (x, y) dominates another point (x', y'), if $x \leq x'$ and $y' \leq y$ with at least one coordinate being strictly smaller. A point set is called a Pareto set if it contains no dominated elements. For a point p, we denote by $p.x$ and $p.y$ the x and y-coordinate of p respectively. The Pareto front of a set S, consists of the non-dominated points in S. We are now ready to define the problems of interest.

▶ **Definition 1** (Bounded Pareto Sum). *Given Pareto sets P, Q of size n with integer coordinates in $[W]^2$, compute the set S of non-dominated points in $P+Q := \{p+q \mid p \in P, q \in Q\}$.*

We remark that for bounded Pareto sum, we always have $W \geq n-1$, as in a Pareto set the coordinates per dimension all need to be different.

▶ **Definition 2** (Bounded Monotone Min-Plus Convolution). *Given two monotone decreasing arrays A and B of size n with positive entries bounded by $W = O(n)$. Compute an array C, where for each $k \in [2n-2]$, we have $C[k] := \min_{i \in [n-1]} (A[i] + B[k-i])$.*

The following reduction is also illustrated in Figure 2 on a small Pareto sum instance[3].

▶ **Theorem 3.** *An instance of bounded Pareto Sum with sets P, Q of size n and entries $(x, y) \in [W]^2$ can be reduced in time $\mathcal{O}(W)$ to an instance of bounded monotone min-plus convolution of two sequences A, B of size W with entries in $[W]$.*

Proof. (Sketch.) We first shift all x-coordinates in P and Q so that the minimum x-value becomes 0; this preserves all dominance relations in the Pareto sum.

Next, we construct two sequences A and B of length W, initialized with value W. For each point $(x, y) \in P$ (resp. $(x, y) \in Q$), we set $A[x] = y$ (resp. $B[x] = y$) and then enforce monotonicity by a single left-to-right sweep: each entry is replaced by the minimum seen so far. After this step, $A[i]$ (resp. $B[i]$) equals the smallest y-value among all points of P (resp. Q) with $x' \leq i$. This construction runs in $\mathcal{O}(W)$ time. Computing the min-plus convolution $C[k] = \min_{i+j=k} A[i]+B[j]$ then yields, for each k, exactly the smallest achievable y-coordinate among all pairs $p \in P$, $q \in Q$ with $p.x + q.x = k$. Thus C contains all candidate points of the Pareto sum. A final linear-time sweep removes dominated entries from C, leaving precisely the non-dominated pairs $(k, C[k])$. Hence the Pareto sum of P and Q can be obtained via a bounded monotone min-plus convolution instance computable in $\mathcal{O}(W)$ time. A more detailed proof can be found in the full version of the paper. ◀

According to our theorem, if bounded monotone min-plus convolution can be solved in time $T(n, W)$ for arrays of size n with entries in $[W]$, then we can solve bounded Pareto Sum in time $O(W + T(W, W))$. This makes algorithms for bounded monotone min-plus convolution applicable to Pareto Sum with little overhead to carry out the reduction and the back-transformation of the solution. By applying the algorithm of [11], we get:

▶ **Corollary 4.** *Given two Pareto sets P, Q of size n with integer entries $(x, y) \in [W]^2$, we can compute the Pareto sum of P and Q in time $\tilde{O}(W^{1.5})$.*

[3] A similar construction has been noted by Chan and Lewenstein for the special case that P and Q are connected sets i.e. all points in P and Q have a neighboring point with L_1 distance 1; see [9, Remark 3.4]

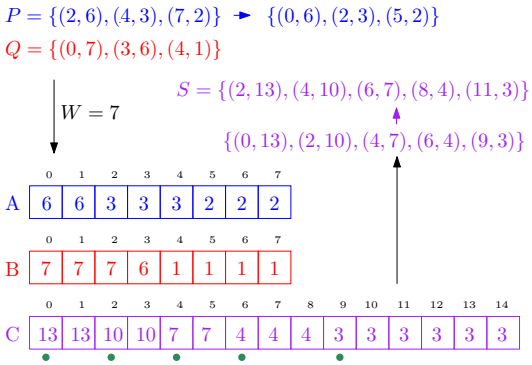


Figure 2 Reduction from Pareto sum to bounded monotone min-plus convolution and back-transformation of the solution on a small example.

3 Additive Approximation of Pareto sum

In this section, we study additive approximation algorithms for Pareto sum with arbitrary real-valued inputs.

To formalize our notion of approximation, observe that the Pareto sum of P and Q can be defined as the set S satisfying the following three conditions: (1) S dominates $P+Q$, i.e.., for every $p \in P$ and $q \in Q$, there exists some $s \in S$ with $s \leq p+q$, (2) $S \subseteq P+Q$ and (3) S is a Pareto set, i.e., pairwise non-dominating. To obtain an *approximate* Pareto set, it is only natural to relax the first condition by allowing an additive slack of $\Delta > 0$ in the domination inequality. Formally, we arrive a the following notion of approximation:

▶ **Definition 5** (Δ-Approximative Pareto sum). *Let $P, Q, \tilde{S} \subseteq \mathbb{R}^2$. We call $\tilde{S}$ a Δ-approximation of the Pareto sum of P, Q if the following conditions hold:*

1. *for all $p \in P, q \in Q$, there exists some $\tilde{s} \in \tilde{S}$ with $\tilde{s} \leq p+q+(\Delta,\Delta)$.*
2. *$\tilde{S} \subseteq P+Q$,*
3. *$\tilde{S}$ is a Pareto set.*

Let us remark that we could have arrived at this definition via slightly adapting the notion of an approximative Pareto front by the authors in [20], which leans on the notion of Papadimitriou and Yannakakis [22]: Let $d_H(S,\tilde{S})$ denote the directed Hausdorff distance between S and $\tilde{S}$ under some "distance" d, i.e., $d_H(S,\tilde{S}) := \max_{s\in S} \min_{\tilde{s}\in\tilde{S}} d(s,\tilde{s})$.

Following [20], we call a set $\tilde{S}$ a Δ-*approximation for the Pareto set* S if $d_H(S,\tilde{S}) \leq \Delta$. As a natural (asymmetric) "distance" measure $d(s,\tilde{s})$ that quantifies the disadvantage of taking $\tilde{s}$ over s (w.r.t domination of other points), we define: $d(s,\tilde{s}) = \max\{0, \tilde{s}.x - s.x, \tilde{s}.y - s.y\}$. Note that this measure is a more suitable definition than using, e.g., $\|\tilde{s}-s\|_\infty$, which would unnaturally punish a point that is worse by $\leq \Delta$ in x-direction, but *much better* (by $\gg \Delta$) in y-direction. With this notion, $\tilde{S}$ is a Δ-approximative Pareto sum of P, Q, if $\tilde{S}$ Δ-approximates S, (2) $\tilde{S} \subseteq P+Q$, and (3) $\tilde{S}$ is a Pareto set.

To obtain such an approximation, we proceed as follows: we first scale down the points of P and Q by dividing the x and y-coordinates by a scaling parameter t, creating the sets P' and Q'. We additionally save a *representative* $\text{rep}(p') \in P, \text{rep}(q') \in Q$ for each (rounded) point $p' \in P'$ and $q' \in Q'$ in the original sets (we choose the one with minimal L_1 norm but any representative works). Then we compute the Pareto sum S' for the sets P' and Q' and determine, from each $s' = p' + q' \in S'$, an approximate point $\tilde{s}$ using the representatives of p' and q', which yields an approximate set $\tilde{S}$. As we shall prove, this simple idea turns out to be conditionally optimal, by devising a fine-grained equivalence with

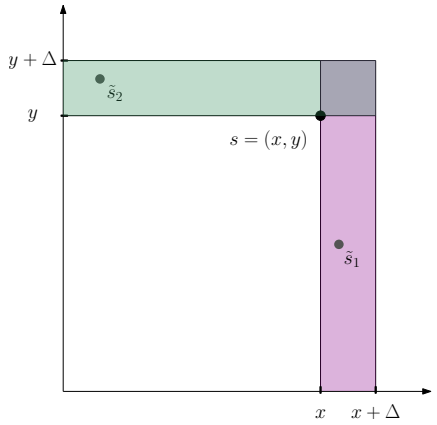


Figure 3 The points $\tilde{s}_1$ and $\tilde{s}_2$ are examples of points with $d(\tilde{s}, s) \leq \Delta$ for the shown $s \in S$. Note that if $s \in S$, then no point in $P + Q$ dominates s (i.e., no point lies to the lower left of s).

bounded $(\min, +)$-convolution. This algorithm is formalized in Algorithm 1, which takes as input P, Q and a scaling factor t, and returns a $2t$-approximation of the Pareto sum of P and Q.

Algorithm 1 ApproximateParetoSum(P, Q, t).

$P' \leftarrow \left\{\left(\lfloor \frac{p.x}{t} \rfloor, \lfloor \frac{p.y}{t} \rfloor\right) \mid p \in P\right\}$
$Q' \leftarrow \left\{\left(\lfloor \frac{q.x}{t} \rfloor, \lfloor \frac{q.y}{t} \rfloor\right) \mid q \in Q\right\}$
for $p' \in P'$ **do**
  $\text{rep}(p') \leftarrow \arg\min_{p \in P} \left\{\|p\|_1 \mid p' = \left(\left\lfloor \frac{p.x}{t} \right\rfloor, \left\lfloor \frac{p.y}{t} \right\rfloor\right)\right\}$
end for
for $q' \in Q'$ **do**
  $\text{rep}(q') \leftarrow \arg\min_{q \in Q} \left\{\|q\|_1 \mid q' = \left(\left\lfloor \frac{q.x}{t} \right\rfloor, \left\lfloor \frac{q.y}{t} \right\rfloor\right)\right\}$
end for
$P' \leftarrow$ Pareto front of P', $Q' \leftarrow$ Pareto front of Q' {*if necessary*}
$S' \leftarrow PS(P', Q')$, where $s' \in S'$ is given as (p', q') with $s' = p' + q'$ and $p' \in P', q' \in Q'$
$Z \leftarrow \{\text{rep}(p') + \text{rep}(q') \mid (p', q') \in S'\}$
return $\tilde{S} \leftarrow$ Pareto front of Z

▶ **Lemma 6.** *For any P, Q and $t > 0$, let S denote $PS(P, Q)$ and let $\tilde{S}$ denote the result of ApproximateParetoSum(P, Q, t). Then $\tilde{S}$ is a $2t$-approximation of the Pareto sum of P, Q.*

Proof. We first argue that the set Z computed in Algorithm 1 satisfies the relaxed domination condition with $\Delta = 2t$, specifically, that for all $p + q$ with $p \in P, q \in Q$, there exists some $z = \text{rep}(p') + \text{rep}(q') \in Z$ with $z \leq p + q + (2t, 2t)$. Indeed, for any $p \in P, q \in Q$, consider their rounded points $p' = \left(\lfloor \frac{p.x}{t} \rfloor, \lfloor \frac{p.y}{t} \rfloor\right) \in P'$, $q' = \left(\lfloor \frac{q.x}{t} \rfloor, \lfloor \frac{q.y}{t} \rfloor\right) \in Q'$. By definition of the Pareto sum S' of P', Q', there must exist some $(\tilde{p} + \tilde{q}) \in S'$ with $\tilde{p} + \tilde{q} \leq p' + q'$. We claim that $z := \text{rep}(\tilde{p}) + \text{rep}(\tilde{q})$ approximately dominates $p + q$, since

$$\begin{aligned} z &= \text{rep}(\tilde{p}) + \text{rep}(\tilde{q}) \leq (t\tilde{p} + (t, t)) + (t\tilde{q} + (t, t)) \\ &= t(\tilde{p} + \tilde{q}) + 2(t, t) \leq t(p' + q') + 2(t, t) \\ &\leq p + q + 2(t, t), \end{aligned}$$

where we used $\text{rep}(x') \leq tx' + (t,t)$ for any rounded point x' in the first line, and $tx' \leq x$ for any x and rounded point x' in the third line. We claim that the lemma follows: Filtering out dominated points in Z ensures that $\tilde{S}$ is a Pareto set (establishing Condition 3.), without affecting the domination condition (if $z \in Z$ dominates $z' \in Z$, then whenever $z' \leq p+q+(t,t)$, then $z \leq z' \leq p+q+(t,t)$, so z' is never needed to ensure the condition). Finally, clearly $\tilde{S} \subseteq Z \subseteq P+Q$ as any element of Z is explicitly given as $\text{rep}(p') + \text{rep}(q')$ for corresponding $\text{rep}(p') \in P$ and $\text{rep}(q') \in Q$. ◀

▶ **Theorem 7.** *Let $T(n,W)$ denote the runtime of monotone min-plus convolution of arrays of size n and entries bounded by W with witness reporting. Given sets P,Q of n points in $\mathbb{R}^2$, we can compute a Δ-approximate Pareto sum in time $\mathcal{O}(n+(W/\Delta)\log(W/\Delta)+T(W/\Delta,W/\Delta))$.*

Proof. Given Lemma 6, it remains to show that $\text{ApproximateParetoSum}(P,Q,t)$ with $t := \Delta/2$ can be computed in the desired running time. After $O(n)$-time preprocessing, the main step in Algorithm 1 is to compute the Pareto sum S' of P', Q'. By our rounding, P', Q' have integer coordinates in $\{0,\dots,\lfloor W/\Delta \rfloor\}$, and Theorem 3 is applicable to obtain S' and thus Z in time $T(\lfloor W/\Delta \rfloor, \lfloor W/\Delta \rfloor)$. Observe that Z is a set of size at most $O(W/\Delta)$. Thus, we can compute its Pareto front in time $O((W/\Delta)\log(W/\Delta))$, concluding the claim. ◀

▶ **Corollary 8.** *Given sets P,Q of n points with real coordinates in $[0,W]^2$ and $\epsilon > 0$, we can compute an ϵW-approximation of the Pareto sum in time $\tilde{\mathcal{O}}\left(n + (1/\epsilon)^{1.6}\right)$.*

Proof. Follows from the $\tilde{O}(n^{1.6})$ time algorithm of [11], and our witness reporting extension, described in the full version of the paper[4]. ◀

Let us also now show a lower bound, which will nicely complement Algorithm 1.

▶ **Theorem 9.** *Let $T_{apx}(n,W,\Delta)$ denote the running time of computing, given Pareto sets P,Q in $\mathbb{R}^2_{\geq 0}$ with maximum coordinate bounded by W, a Δ-approximation of their Pareto sum, with witnesses. If there exists $\alpha \geq 0$ and c such that $T_{apx}(n,W,\Delta) \leq O(n+(W/\Delta)^c)$ whenever $\Delta = \Theta(W^\alpha)$, then bounded monotone min-plus convolution can be computed in time $O(n^c)$, with witnesses.*

Proof. The proof is an adaptation of the reduction from min-plus convolution to Pareto sum given in [13]. Consider an arbitrary min-plus convolution instance $A[0,...,n-1], B[0,...,n-1]$ with monotonicity (specifically, $A[i] > A[i+1], B[i] > B[i+1]$ for all i) and entries in $\{0,\dots,O(n)\}$. We construct the sets of points

$$P := \{(i \cdot (2n^{\frac{\alpha}{1-\alpha}}), A[i] \cdot (2n^{\frac{\alpha}{1-\alpha}})) \mid i \in [n-1]\},$$
$$Q := \{(i \cdot (2n^{\frac{\alpha}{1-\alpha}}), B[i] \cdot (2n^{\frac{\alpha}{1-\alpha}})) \mid i \in [n-1]\}.$$

Consider a Δ-approximation $\tilde{S}$ of the Pareto sum of P,Q with $\Delta = n^{\frac{\alpha}{1-\alpha}}$. By construction of the point sets, if a point $\tilde{s} \in \tilde{S} \subseteq P+Q$ Δ-dominates a point $p+q$, i.e., $\tilde{s} \leq p+q+(\Delta,\Delta)$, then it already dominates $p+q$, i.e., $\tilde{s} \leq p+q$. Thus, $\tilde{S}$ is the true Pareto sum of P,Q, and we can extract the min-plus convolution C of A,B: By construction, for every $k \in [2n-2]$, there must be a point in $PS(P,Q)$ with x-coordinate $k \cdot (2n^{\frac{\alpha}{1-\alpha}})$ and y-coordinate $(\min_{i \in [n-1]} A[i] + B[k-i]) \cdot (2n^{\frac{\alpha}{1-\alpha}})$, revealing $C[k] = \min_{i \in [n-1]}(A[i]+B[k-i])$. The existence of a point with x-coordinate $k \cdot (2n^{\frac{\alpha}{1-\alpha}})$ for every $k \in [2n-2]$ follows by the decreasing values of $C[k]$. Also, witnesses for such points can be transferred.

[4] We remark that it is very plausible that our witness reporting algorithm extends to the $\tilde{O}(n^{1.5})$ algorithm.

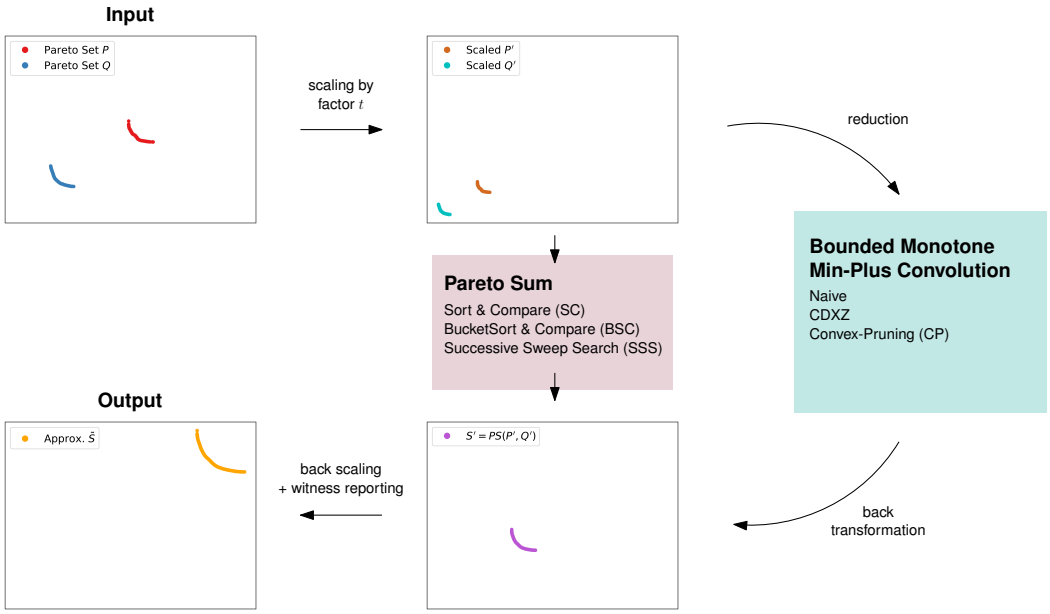


Figure 4 Overview of the main steps of our additive approximation algorithms. An important step is to compute the Pareto sum of the scaled instance. This can either be done directly via a Pareto sum algorithm or by an algorithm for bounded monotone min-plus convolution when applying the proper forward and backward reduction steps.

Note that the constructed instance of Pareto sum Approximation consists of n points in P, Q with coordinates bounded by $W \leq O(n^{1+\frac{\alpha}{1-\alpha}}) = O(n^{\frac{1}{1-\alpha}})$ and $\Delta = n^{\frac{\alpha}{1-\alpha}}$. Thus, if $T_{apx}(n, W, \Delta) \leq O(n + (W/\Delta)^c) = O(n + (n^{\frac{1}{1-\alpha}} - \frac{\alpha}{1-\alpha})^c)$, then we can compute the min-plus convolution of A, B in time $O(n^c)$, with witnesses. ◀

▶ Remark 10. Even without witness reporting, we still get a fine-grained equivalence (for slightly weaker guarantees), by a minor adaptation of Algorithm 1 and Theorem 9. We relax the condition $\tilde{S} \subseteq P + Q$ to the milder condition that each $\tilde{s}$ is in the vicinity of a point $p + q \in P + Q$. More formally, for all $\tilde{s} \in \tilde{S}$ there exist $p \in P$ and $q \in Q$ such that $p + q \leq \tilde{s} \leq p + q + 2(t, t)$. Further details can be found in the full version of the paper.

4 Framework and Implementation

Our proposed additive approximation algorithm, presented in Algorithm 1, consists of scaling down the instance, computing the Pareto sum of the transformed point sets, and then scaling up the solution. The Pareto sum instance after scaling is captured by our definition of bounded Pareto sum. To compute its solution, we can obviously directly apply any Pareto sum algorithm. By virtue of Theorem 3, however, also any algorithm for bounded min-plus convolution translates to an algorithm for bounded Pareto sum when performing the required reduction and back-transformation of the solution. Thus, depending on whether we use an algorithm directly designed for Pareto sum or a convolution algorithm, the computation pipeline needs to be adapted. Figure 4 provides an overview of our framework. In the following, we describe the algorithms of each category in more detail.

4.1 Algorithms for (bounded) Pareto sum

We start with the algorithms designed directly for Pareto sum computation.

Sort & Compare (SC) and Successive sweep search (SSS)

We implemented both state of the art algorithms for (general) Pareto sum computation from [13] as a baseline. In a nutshell, the Sort & Compare algorithm sorts the points in the sumset $P + Q$ lexicographically and performs constant time dominance checks per point based on this ordering. Further details on the baselines can be found in the full version of the paper.

BucketSort & Compare (BSC)

For bounded integral Pareto sum, we can improve the Sort & Compare algorithm, by using bucketsort to sort the points, and then perform a direct update on the minimal y-coordinate for each corresponding x-coordinate. Further details can be found in the full version of the paper.

4.2 Algorithms for min-plus convolution

After the reduction to min-plus convolution, we can of course use the naive quadratic time algorithm for the latter. In the following, we discuss two more sophisticated and (potentially) faster convolution algorithms.

CDXZ Algorithm

We discuss a simplified version of the $\tilde{O}(n^{1.6})$-algorithm [11, Section 4.1] for monotone bounded min-plus convolution in a detailed fashion in the full version of the paper. Using this algorithm in our Pareto sum reduction framework results in a running time of $\mathcal{O}(W^{1.6} \log W)$ for bounded Pareto sum. Let us highlight the main improvements and simplifications we achieved.

- We removed the segment tree data structure. Making use of a simpler enhanced min-plus convolution algorithm instead of using a segment tree with interval updates is simpler and improves the runtime theoretically and practically.
- We improved the bound of b, which was previously ranging from $b \in \{0, 1, 2\}$ to $b \in \{0, 1\}$. The better bound of the constant b, is insignificant in the theoretical analysis of the algorithm, but it plays an important role in the error correction part. Besides the initial polynomial multiplication of P and Q, every other part of the algorithm in the error correction phase will be repeated b-times, thus the better bound on b will result on repeating these costly parts of the algorithm twice instead of thrice.
- As the set T_b can be of expected size $\mathcal{O}(n^{1.6})$ (for the choice of $\beta = 0.4$), we do not save it explicitly like [11] suggests. In order to circumvent storing the set T_b, we make an online computation of the polynomials $R_{p,b}[k](x)$ simultaneously, updating the polynomials after having seen a new pseudo-witness $(i, k - i)$ on the fly.

We implemented the version without witness reporting, as the witness reporting algorithm would be too inefficient in practice. This leads to a weak approximation of the Pareto sum. For the multivariate polynomial multiplication, we perform a standard reduction to the univariate case, by turning P and Q into univariate polynomials. Instead of FFT, we make use of the polynomial multiplication offered by the library FLINT [24], where the univariate

polynomials are kept as lists. Instead of using binary search trees in the computation of T_b, we performed a linear search, after saving for each possible modulo value $i \in [p-1]$ the indices k with $C'[k] + b \mod p = i$. This turned out to be faster in the experiments. The parameters α and β are not fixed as the optimal theoretical runtime suggests. The choice of the parameters α and β are transferred to the choice of the scaling factor n^α and the prime p, which are now inputs of the implementation; see Figure 7 for the influence of these parameters on the runtime.

Convex Pruning (CP) algorithm

Bringmann and Cassis developed an algorithm for min-plus convolution, suited towards the case, where the arrays A and B are near-convex [5]. We implemented two versions of their algorithm with and without witness reporting. Our implementation is close to the descriptions in [5], with the main difference being that we did not use sparse convolutions, but rather an enhanced min-plus convolution computation, which works particularly well when the arrays A and B have few distinct entries. We precomputed groups of entries of the same value in A and B. More details on the enhanced min-plus convolution on the precise changes can be found in the full version of the paper.

4.3 Further engineering

Algorithm 1 has been implemented mostly matching its description, with small changes, which we describe now. When using a convolution algorithm within Algorithm 1, we also perform the needed reduction from bounded Pareto sum to monotone bounded min-plus convolution and the transformation backwards. The reduction closely follows the constructive proof of Theorem 3. At the end of Algorithm 1, instead of computing the Pareto front of the set Z by sorting, we do the following: For each newly inserted point, we look back and remove all points, which are dominated by this new point. This will work fast as the set Z is almost a Pareto set, so few corrections need to be made.

5 Experimental Evaluation

In the following, we evaluate our approximate algorithm with the different choices for bounded Pareto sum computation sketched above. While all algorithms are described for the case that P and Q are of the same size n, they also all work for differing sizes. Our code is written in C++. It was compiled with gcc using -O3. In our implementation of the CDXZ algorithm, we use routines from FLINT (Fast Library for Number Theory) [24]. All experiments were conducted on a single core of an AMD Threadripper 3970 with 256 GB of RAM. We set a time-out of one hour per run. Running times are always averaged over 10 runs.

Benchmarks. We use three types of benchmarks in our evaluation. Two are synthetic to allow us to control instance properties, while the third benchmark consists of real data.

Benchmark 1: Instances with bounded range. First, we use an instance generator that allows us to fix n and W. Here, it is important to ensure $W \geq n-1$ to guarantee that there are enough distinct natural numbers in the range to construct a Pareto set. We use $W = c \cdot n$ for $c \geq 1$. To efficiently create a strictly monotonically increasing sequence, we initialize an array with a counter γ_i for each $i \in [W]$. Then, we increment a randomly selected counter in each of n rounds. Afterwards, we sweep over this array, starting at position 0, and perform

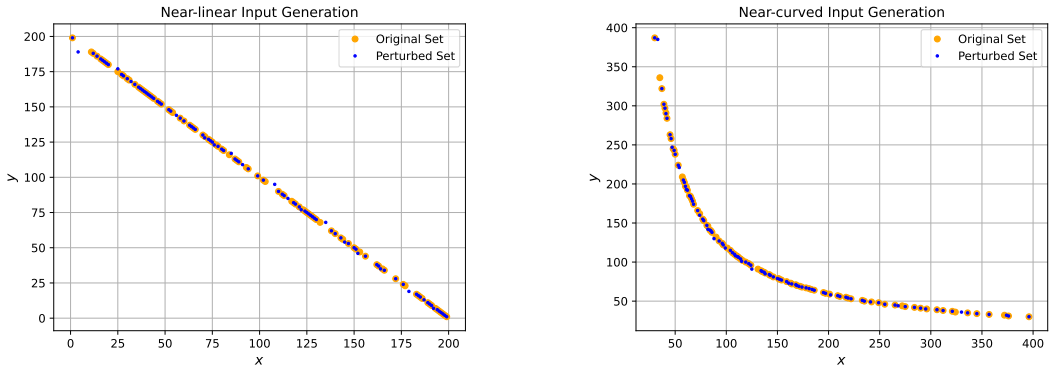


Figure 5 Two different versions of generating input sets with values in the range $[0, 2n]$: Near-linear and near-curved input sets are based on the linear and curve generator, respectively, where we apply perturbation to a subset of the points.

the following operations: If $\gamma_i > 1$, we set $\gamma_i = 1$ and $\gamma_{i+1} = \gamma_{i+1} + \gamma_i - 1$. Here, entry 0 is assumed to be the successor of γ_W. We stop when all entries are 0 or 1 which happens after each entry has been considered at most twice. Then, for all $\gamma_i = 1$, element i is added to the sequence. This produces a valid sequence in time $\mathcal{O}(n + W)$. Pareto sets can then be constructed by using two such sequences, one sorted increasingly for the x-coordinates and the other one decreasingly for the corresponding y-coordinates.

Benchmark 2: Function-based instance generator. For any strictly monotonically decreasing function f defined on $\mathbb{R}^+$, sampling n points $(x, f(x))$ provides us with a Pareto set. We use two base functions: the first uses $f(x) = -x$ and the second $f(x) = \frac{c}{x}$ with a constant $c \in \mathbb{R}^+$. We slightly perturb the points to have more input variability, while guaranteeing that pairwise non-dominance is upheld. We refer to the resulting instances as near-linear or near-curved. See Figure 5 for an illustration. We note that by means of the perturbation, the resulting point sets are no longer convex.

Benchmark 3: Route planning instances. Finally, we also use real-world data sets for evaluation. The instances stem from bi-criteria route planning in line with [13]. Here, given a graph in which the edges are augmented with two cost values (in our case distance in meters and positive height difference in centimeters), the goal is to compute all Pareto-optimal paths from a source node s to a target node t. Using bi-directional search to answer such queries, or constructing a data structure which accelerates query answering [23], demands to combine and filter the set of Pareto-optimal paths from s to some intermediate node v, and from v to the target t. The cost pairs of these two path sets form our inputs P and Q. The benchmark contains roughly 100000 instances based on randomly chosen s, v and t triples in a directed road network with one million nodes and two million edges. Note that the sizes of P and Q might differ significantly in each instance.

Results for Exact Pareto Sum. For a general comparison of the different algorithms for Bounded Pareto sum discussed in this paper, we first applied a scalability study on the range-bound instances with all point coordinates being integers, see Figure 6. We observe huge performance differences. SSS is by far the slowest approach. Its theoretical performance is only subquadratic in case the output size is in $o(n)$. However, for the considered instances there are roughly $4n$ output points. The BSC algorithm performs best and also significantly

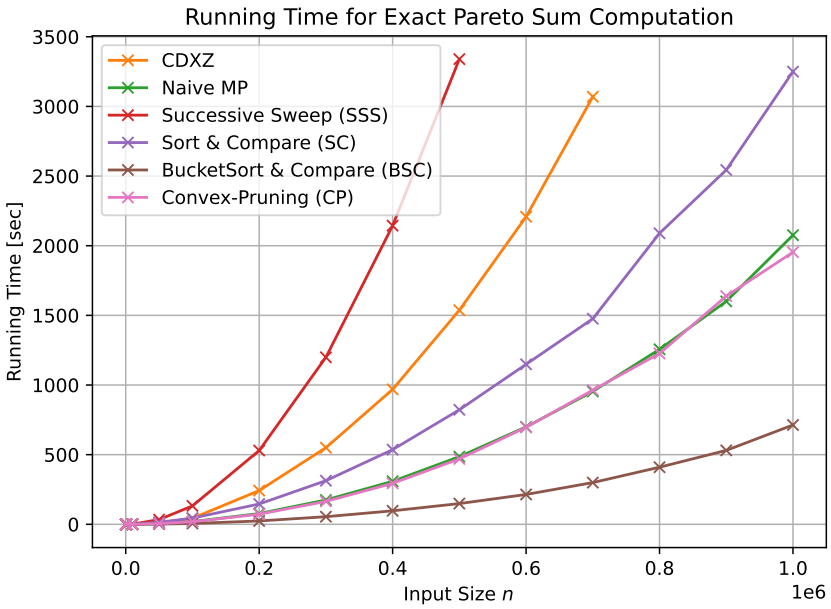


Figure 6 Running times for Pareto sum computation on generated inputs of size up to $n = 100,000$ with values in the range of $[0, 2n]$ and a timeout of 1 hour.

outperforms the plain SC algorithm. Even more remarkably, also the min-plus convolution based algorithms, naive and CP, outperform SC. While CDXZ beats SSS, it is a bit slower than SC on this instance type. As described in Section 4, our implementation of the CDXZ algorithm allows us to choose its parameters. We conducted a series of tests on synthetic instances of different sizes to determine a good parameter choice, see Figure 7 for an example. We observe that the parameter values heavily impact the performance. Across all tested

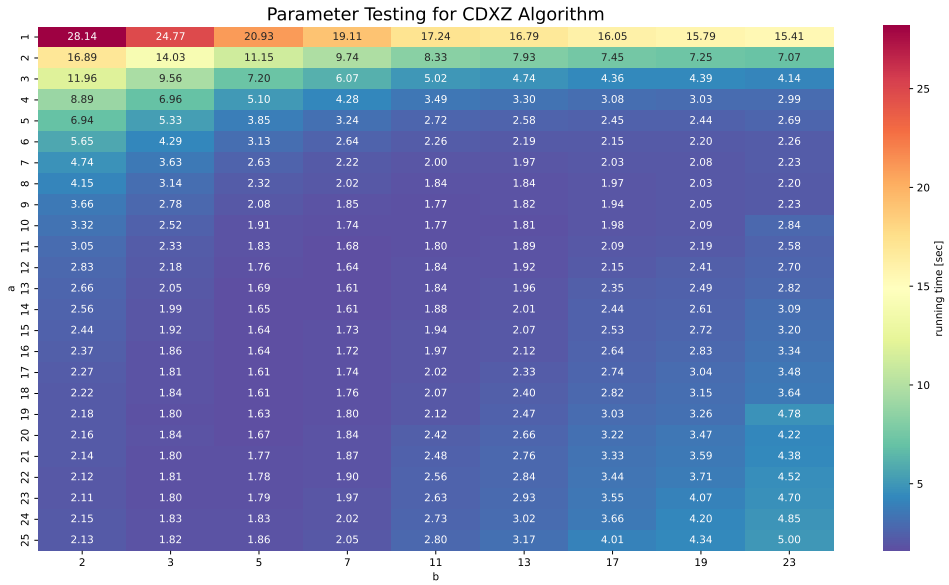


Figure 7 Parameter testing for the CDXZ algorithm on inputs of size $n = 20000$ in the range $2n$.

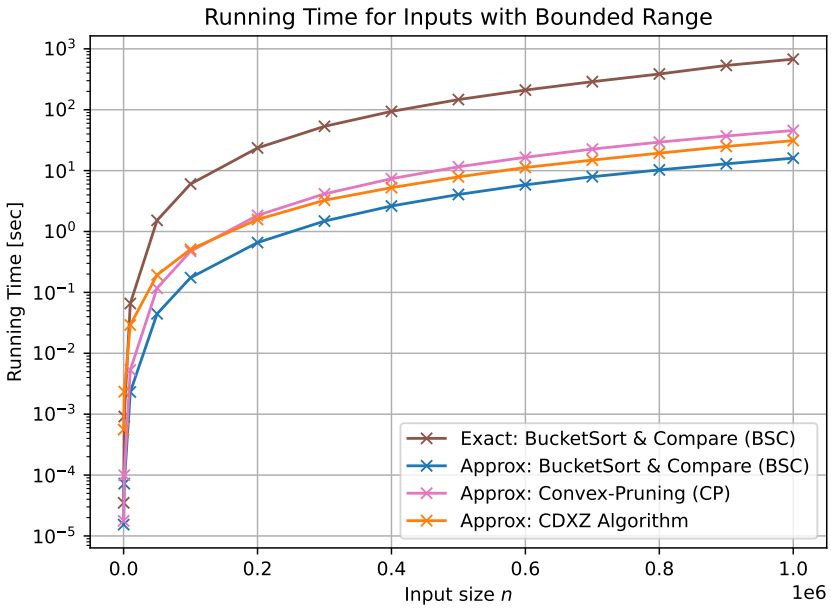


Figure 8 Running time of the approximation variants on inputs generated with bounded range of $2n$ using a scaling factor of $t = 10$.

instances, scaling factors of 25 and $p = 2$ were among the top configurations. We therefore use those values in all subsequent experiments. We remark, though, that additional fine-tuning or choices adaptive to n could potentially further improve the performance.

However, our main interest is in computing approximate solutions. In the following, we investigate how the algorithms perform when used as a plug-in in our approximation framework. In the remainder of the experiments, we focus on the BSC algorithm as well as the convolution based algorithms, as they outperform the baseline algorithms.

Results for Approximate Pareto Sum. Next, we present comparative running time results of our approximation algorithm with different inner methods of Pareto sum computation applied to the scaled down inputs. For fair comparison, we use the non-witness reporting variants for all of them. Figures 8, 9, and 10 depict the results for range-bounded, near-linear and near-curve instances, respectively, using a value range of $2n$ and a scaling factor of $t = 10$. Interestingly, for each input type, another algorithm performs best. For range-bounded instances, BSC is the fastest with CDXZ being second, while on near-linear instances CDXZ outperforms the other two methods. On near-curved instances, however, CP is clearly the best, beating the others by at least an order of magnitude. The performance of CP relies on its ability to prune pairs, which on near-curved instances applies to up to 95% of total point pairs, see Figure 11.

Finally, we consider our third benchmark set, which consists of bi-criteria route planning instances of varying size. Figure 13 shows the running time and approximation quality of our algorithms for different scaling factors. We observe that BSC performs best with CP being a close second, especially for larger values of t. They both improve on exact computation by over an order of magnitude and also stay well below the theoretical additive approximation bound of $2t$. Examples of approximate solutions can be found in Figures 14. One can see that even when choosing a larger t, the Pareto sum is still well represented by the computed

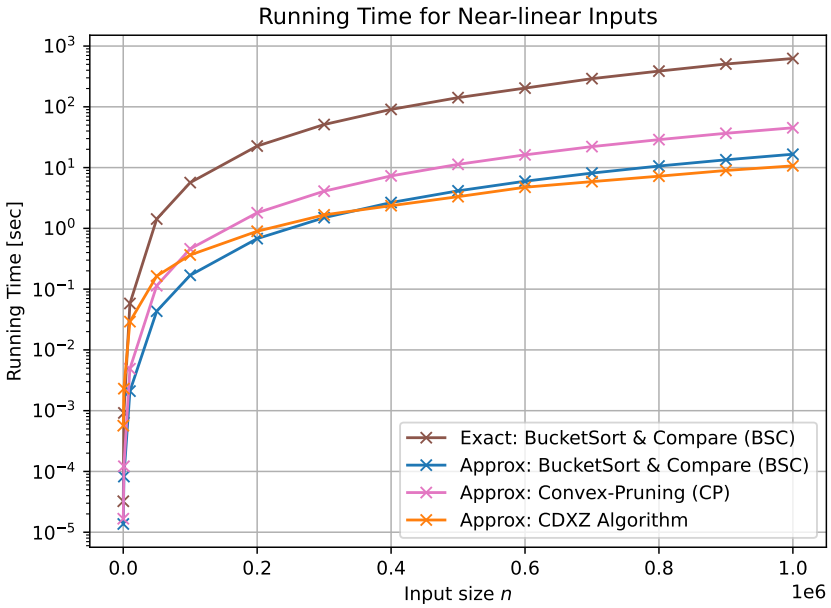


Figure 9 Running time of the approximation variants with a scaling factor of $t = 10$ on near-linear inputs generated in the range of up to $2n$.

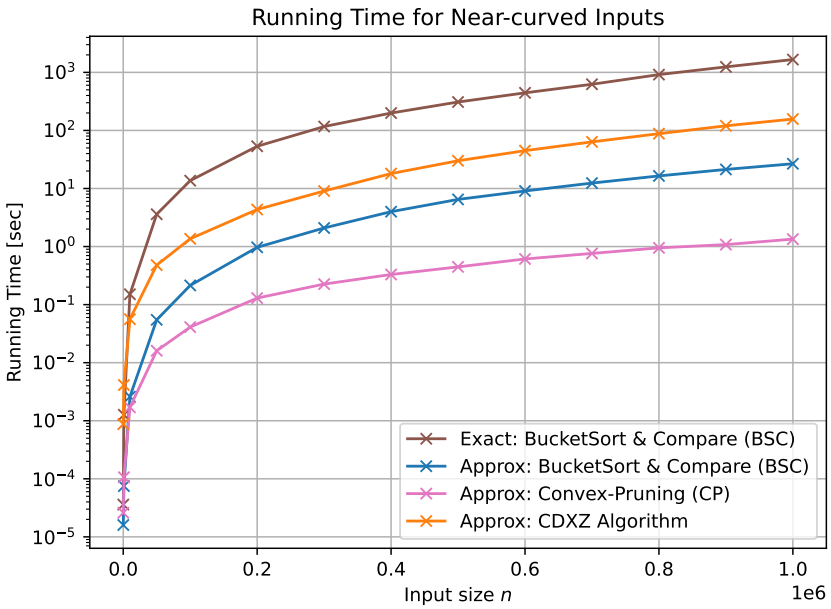


Figure 10 Running time of the approximation variants with a scaling factor of $t = 10$ on near-curved generated inputs in the range of up to $2n$.

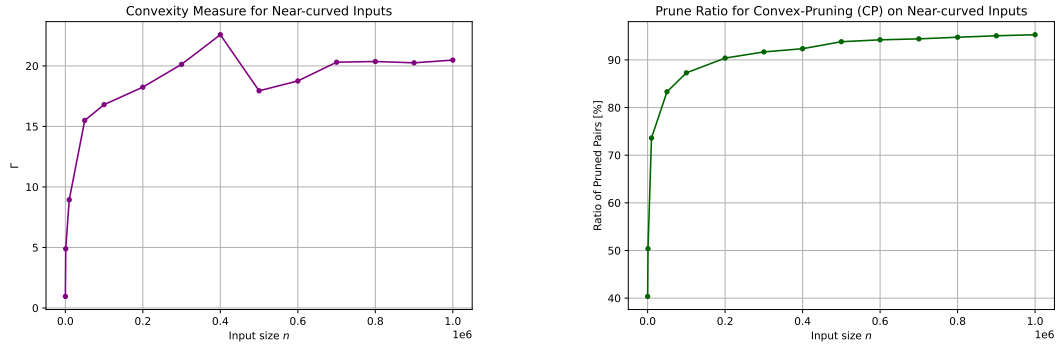


Figure 11 Convexity measure for near-curved inputs (left) and ratios of pruned pairs by the CP algorithm with respect to all available pairs in the index space (right).

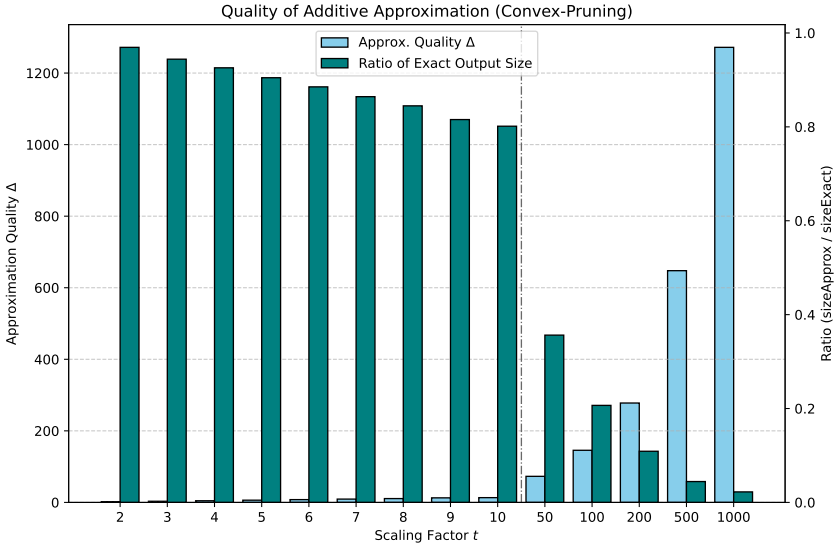


Figure 12 Quality of the approximation using the CP algorithm on real benchmarks for varying scaling factors $t = 2, \ldots, 1000$. We compare the ratio of the output size of the approximation and the exact algorithm to the approximation quality Δ.

solution. Indeed, larger values of t are quite reasonable in our application. For example, a route that is up to 100 meters longer than another one is almost indistinguishable to a user if the total route length is anyway in the orders of kilometers.

Furthermore, the user usually does not want to be presented with hundreds of routing options, but a concise and diverse set of reasonable options. This is achievable with our approach, as we can not only reduce the running time with larger t but also the output size, see Figures 15 and 12. Clearly, our algorithm provides good trade-offs between running time and solution quality, which allows one to choose the best t depending on the application.

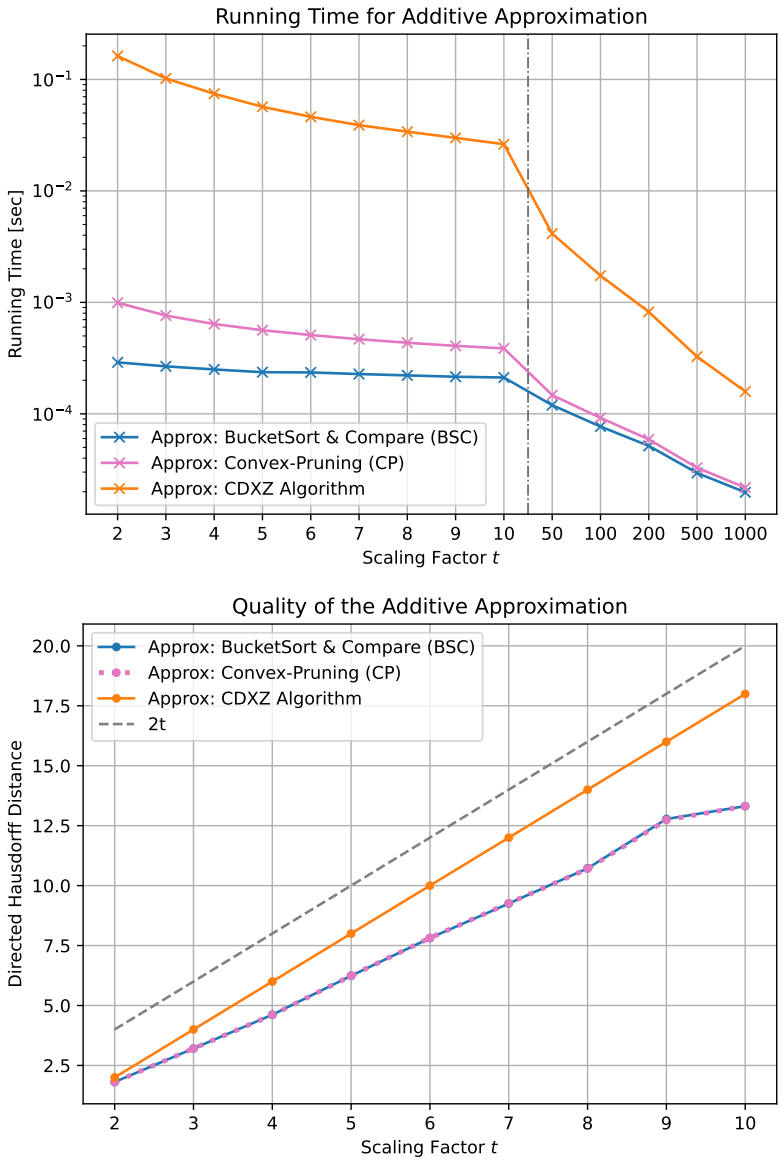


Figure 13 Running time of approximations on real benchmark for varying scaling factors $t = 2, \ldots, 1000$ (top) and solution quality for $t = 2, \ldots, 10$ (bottom). This shows the difference of the quality of approximation with (BSC,CP) and without witness reporting(CDXZ).

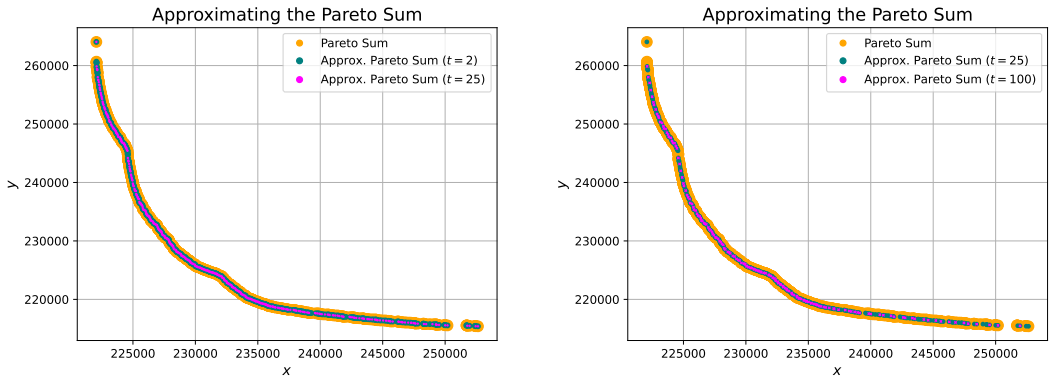


Figure 14 Example for approximate Pareto sum on real benchmark data. The approximative solution for $t = 2$ results in a quality of $\Delta = 2$ (left) whereas for $t = 25$ the quality is $\Delta = 38$ and for $t = 100$ we have $\Delta = 153$ (right).

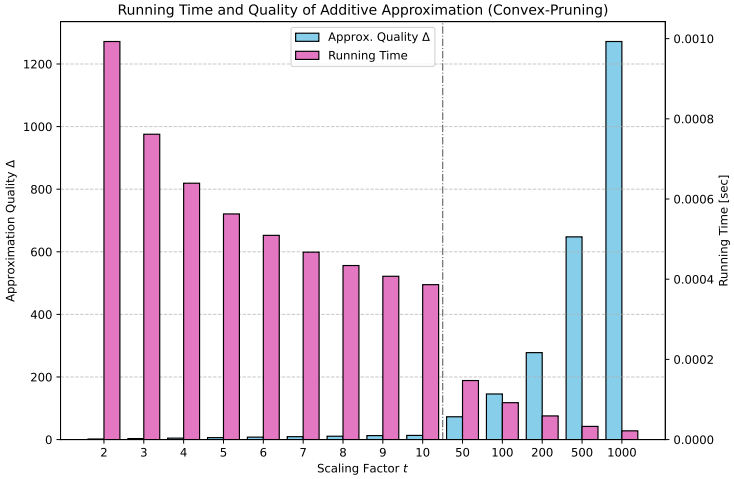


Figure 15 Approximation quality Δ and running time of the approximation using the CP algorithm on real benchmarks with varying scaling factor $t = 2, \ldots, 1000$.

6 Future Work & Discussion

We have established that fine-grained reductions are not only a means to obtain theoretical intractability results, but also a tool to efficiently solve problems via reductions to other problems for which fast algorithms are implemented and engineered. To further foster (bounded monotone) min-plus convolution as a core primitive, it would be interesting to know whether the $\tilde{O}(n^{1.5})$ algorithm [11, Section 4.2], admits an efficient implementation. The main drawback right now is that it needs 9 calls to the monotone bounded min-plus function, and that the constant $b \in [-4, 7]$ is much larger than in the algorithm we implemented. We believe that the CP algorithm can benefit from practical sparse convolution algorithms, especially on near-linear inputs. Is it possible to engineer fast sparse convolution algorithms in practice? Finally, can we leverage our min-plus convolution results to obtain practical improvements for knapsack [4, 6] or the tree sparsity problem [21, 2]?

References

1 Noga Alon and Moni Naor. Derandomization, witnesses for boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16(4/5):434–449, 1996. doi:10.1007/BF01940874.

2 Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. Better approximations for tree sparsity in nearly-linear time. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2215–2229. SIAM, 2017. doi:10.1137/1.9781611974782.145.

3 Karl Bringmann. Approximating subset sum ratio faster than subset sum. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 1260–1277. SIAM, 2024. doi:10.1137/1.9781611977912.50.

4 Karl Bringmann and Alejandro Cassis. Faster knapsack algorithms via bounded monotone min-plus-convolution. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 31:1–31:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.31.

5 Karl Bringmann and Alejandro Cassis. Faster 0-1-knapsack via near-convex min-plus-convolution. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, Amsterdam, The Netherlands, September 4-6, 2023*, volume 274 of *LIPIcs*, pages 24:1–24:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.24.

6 Karl Bringmann, Anita Dürr, and Adam Polak. Even faster knapsack via rectangular monotone min-plus convolution and balancing. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 33:1–33:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.33.

7 Karl Bringmann, Ahmed Ghazy, and Marvin Künnemann. Exploring the Approximability Landscape of 3SUM. In Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2024.34.

8 Karl Bringmann and Vasileios Nakos. A fine-grained perspective on approximating subset sum and partition. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1797–1815. SIAM, 2021. doi:10.1137/1.9781611976465.108.

9 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3sum via additive combinatorics. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 31–40. ACM, 2015. doi:10.1145/2746539.2746568.

10 Wei-Mei Chen, Hsien-Kuei Hwang, and Tsung-Hsi Tsai. Maxima-finding algorithms for multidimensional samples: A two-phase approach. *Computational Geometry*, 45(1-2):33–53, 2012. doi:10.1016/J.COMGEO.2011.08.001.

11 Shucheng Chi, Ran Duan, Tianle Xie, and Tianyi Zhang. Faster min-plus product for monotone instances. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1529–1542. ACM, 2022. doi:10.1145/3519935.3520057.

12 Weiming Feng and Ce Jin. Approximately counting knapsack solutions in subquadratic time. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 1094–1135. SIAM, 2025. doi:10.1137/1.9781611978322.32.

13 Daniel Funke, Demian Hespe, Peter Sanders, Sabine Storandt, and Carina Truschel. Pareto sums of pareto sets: Lower bounds and algorithms. *Algorithmica*, pages 1–34, 2025.

14 Geri Gokaj and Marvin Künnemann. Completeness theorems for k-sum and geometric friends: Deciding fragments of linear integer arithmetic. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, January 7-10, 2025, Columbia University, New York, NY, USA*, volume 325 of *LIPIcs*, pages 55:1–55:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ITCS.2025.55.

15 Geri Gokaj, Marvin Künnemann, Sabine Storandt, and Carina Truschel. (multivariate) k-sum as barrier to succinct computation. In Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman, editors, *33rd Annual European Symposium on Algorithms, ESA 2025, September 15-17, 2025, Warsaw, Poland*, volume 351 of *LIPIcs*, pages 42:1–42:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ESA.2025.42.

16 Boris Goldin and Oren Salzman. Approximate bi-criteria search by efficient representation of subsets of the pareto-optimal frontier. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pages 149–158, 2021. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/15957.

17 David G Kirkpatrick and Raimund Seidel. Output-size sensitive algorithms for finding maximal vectors. In *Proceedings of the First Annual Symposium on Computational Geometry*, pages 89–96, 1985. doi:10.1145/323233.323246.

18 Kathrin Klamroth, Bruno Lang, and Michael Stiglmayr. Efficient dominance filtering for unions and minkowski sums of non-dominated sets. *Computers & Operations Research*, 163:106506, 2024. doi:10.1016/J.COR.2023.106506.

19 Joshua Marc Könen, Heiko Röglin, and Tarek Stuck. Parameterized Algorithms for Computing Pareto Sets. In Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman, editors, *33rd Annual European Symposium on Algorithms (ESA 2025)*, volume 351 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 105:1–105:15, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2025.105.

20 Julien Legriel, Colas Le Guernic, Scott Cotton, and Oded Maler. Approximating the pareto front of multi-criteria optimization problems. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2010. doi:10.1007/978-3-642-12002-2_6.

21 Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. A subquadratic approximation scheme for partition. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 70–88. SIAM, 2019. doi:10.1137/1.9781611975482.5.

22 Christos H. Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, Redondo Beach, California, USA, November 12-14, 2000*, pages 86–92. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892068.

23 Sabine Storandt. Route planning for bicycles—exact constrained shortest paths made practical via contraction hierarchy. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 22, pages 234–242, 2012.

24 The FLINT team. *FLINT: Fast Library for Number Theory*, 2025. Version 3.3.1, https://flintlib.org.

25 Carina Truschel and Sabine Storandt. Multi-Criteria Route Planning with Little Regret. In Jonas Sauer and Marie Schmidt, editors, *25th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2025)*, volume 137 of *Open Access Series in Informatics (OASIcs)*, pages 13:1–13:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ATMOS.2025.13.

Linear Time Single-Source Shortest Path Algorithms in Euclidean Graph Classes

Joachim Gudmundsson
The University of Sydney, Australia

Yuan Sha
The University of Sydney, Australia

Sampson Wong
The University of Copenhagen, Denmark

Abstract

In the celebrated paper of Henzinger, Klein, Rao and Subramanian (1997), it was shown that planar graphs admit a linear time single-source shortest path algorithm. Their algorithm unfortunately does not extend to Euclidean graph classes. We give criteria and prove that any Euclidean graph class satisfying the criteria admits a linear time single-source shortest path algorithm. As a main ingredient, we show that the contracted graphs of these Euclidean graph classes admit sublinear separators.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Graph algorithms, Single-Source Shortest Path, Euclidean Graphs, Recursive Division

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.55

Related Version *Full Version*: `http://arxiv.org/abs/2603.22948`

Funding *Joachim Gudmundsson*: Supported by the Australian Government through the Australian Research Council (project number DP240101353).
Yuan Sha: Supported in part by the the Australian Research Council project number DP240101353.
Sampson Wong: Supported by the European Union's Marie Skłodowska-Curie Actions Postdoctoral Fellowship, grant number 101146276.

Acknowledgements We would like to thank David Eppstein for his helpful discussions.

1 Introduction

Computing shortest paths is a fundamental problem in graph theory and in network analysis. Given a graph with non-negative edge weights, the single-source shortest path (SSSP) problem is to compute the shortest paths from a source vertex to all the other vertices in the graph. The SSSP problem has found numerous applications in transportation networks, geographic information systems, motion planning, social networks, chip design, and so on.

The most well-known SSSP algorithm is Dijkstra's algorithm [2]. Using a Fibonacci heap [8], Dijkstra's algorithm runs in $O(m + n \log n)$ time on graphs with m edges and n vertices. Dijkstra's algorithm has a lower bound of $\Omega(n \log n)$ in the comparison computer model, since the algorithm sorts the vertices of the graph by their distances from the source vertex.

In planar graphs, faster algorithms are known. Frederickson [7] gave an $O(n\sqrt{\log n})$ time SSSP algorithm for planar graphs, using *r-division* and graph contraction. An r-division of a planar graph divides the graph into $O(n/r)$ regions such that each region has $O(r)$ vertices and $O(\sqrt{r})$ boundary vertices. Henzinger, Klein, Rao and Subramanian [12] use *recursive division*, a recursive version of r-division, and an edge-relaxation algorithm on the recursive


© Joachim Gudmundsson, Yuan Sha, and Sampson Wong;
licensed under Creative Commons License CC-BY 4.0

division to give an optimal linear-time SSSP algorithm for planar graphs. The algorithms leverage two key properties of planar graphs: (i) the graph class is minor-closed [13], and (ii) the graphs admit a sublinear separator that can be computed in linear time [16]. Tazari and Müller-Hannemann [23] showed how to use knitted H-partition to avoid introducing arbitrarily large minors and extend the linear time algorithm of HKRS [12] to any minor-closed graph class. A natural open problem is to ask: which other graph classes admit a linear time SSSP algorithm?

▶ **Open Problem 1.** *Other than planar graphs and minor-closed graphs, which graph classes admit a linear time single-source shortest path algorithm?*

Practical applications have motivated researchers to study Open Problem 1. For example, road networks are non-planar, and it is unknown if road networks are minor-closed, yet computing shortest paths remains a fundamental problem on road networks.

Le and Than [15] proposed τ-lanky graphs, which are any Euclidean graph where any ball in $\mathbb{R}^d$ with radius r cuts at most τ edges longer than r. Lanky graphs unify several Euclidean graph classes including greedy spanners in $\mathbb{R}^d$ (any fixed d), low-density graphs in $\mathbb{R}^d$ ([1]) and greedy spanners of unit ball graphs in $\mathbb{R}^d$. All these graph classes are $O(1)$-lanky graphs. The low-density graph in $\mathbb{R}^2$ is argued to be a realistic model for road networks [1].

In the conference version of their paper, Le and Than [15] proved that lanky graphs in $\mathbb{R}^d$ admit sublinear separators. They claimed that their (expected) linear time sublinear separator algorithm for lanky graphs implies a linear time SSSP algorithm by applying HKRS's algorithm [12]. Unfortunately, this claim has a gap. While lanky graphs admit sublinear separators, they are not minor-closed. Therefore, HKRS's algorithm does not directly extend to lanky graphs. The SSSP claim is retracted in the full version of the paper [14]. Prior to their paper, Eppstein and Khodabandeh [4] studied the edge crossing patterns of greedy spanners in $\mathbb{R}^2$. They gave an $O(n \log^{(i)} n)$ time SSSP algorithm for greedy spanners in $\mathbb{R}^2$ (Corollary 21), where i is any constant and $\log^{(i)} n$ denotes the i-times iterated logarithm. These efforts raise the following open problem.

▶ **Open Problem 2.** *Is there a linear time SSSP algorithm for τ-lanky graphs in $\mathbb{R}^d$?*

Miller, Teng, Thurston and Vavasis [21, 18, 19] developed a geometric characterization of graphs that have a sublinear separator, using the notion of a *k-ply neighborhood system*. A k-ply neighborhood system is a collection of balls in $\mathbb{R}^d$ such that no point in the space is covered by more than k of the balls. They showed that (the intersection graphs of) k-ply neighborhood systems can model sphere-packings [18], k-nearest neighbor graphs [18] and finite element meshes [20]. They proved that k-ply neighborhood systems in $\mathbb{R}^d$ (and their intersection graphs) admit a sublinear separator that can be computed in linear time [18, 5].

Eppstein and Goodrich [3] use k-ply neighborhood system in $\mathbb{R}^2$ as a model of real-world road networks. They showed how to compute SSSP in randomized linear time, assuming the arrangement of the k-ply disk neighborhood system is given. Besides the assumption, their approach is specific to $\mathbb{R}^2$ and does not extend to higher dimensions. This raises the following open problem.

▶ **Open Problem 3.** *Is there a linear time SSSP algorithm for the intersection graph of a k-ply neighborhood system in $\mathbb{R}^d$, where $d \geq 2$?*

Smith and Wormald [22] considered a collection of cubes in $\mathbb{R}^d$ such that no point in the space is covered by more than κ of the cubes. Here we call the collection of cubes a *κ-thick cubical neighborhood system*. The authors proved that κ-thick cubical neighborhood systems in $\mathbb{R}^d$ (and their intersection graphs) admit a sublinear separator that can be computed in randomized linear time. We consider the following open problem.

▶ **Open Problem 4.** *Is there a linear time SSSP algorithm for the intersection graph of a κ-thick cubical neighborhood system in $\mathbb{R}^d$?*

1.1 Our contribution

Our main technical contribution is to compute a recursive division in linear time, for any Euclidean graph class that satisfies the following **criteria**:

- **(I)** The graph G admits a sublinear separator that can be computed in linear time.
- **(II)** The combinatorially contracted graphs G_i of G (refer to Definition 8) are sparse, i.e., any $\bar{k}$-vertex subgraph of G_i is sparse.
- **(III)** A closed surface in $\mathbb{R}^d$ (such as a Jordan curve in $\mathbb{R}^2$, a sphere in $\mathbb{R}^d$) is used to find the separator in (I). The closed surface cuts a sublinear number of geometrical objects (such as segments, balls) associated with the vertices in G, and these associated vertices form the separator. Moreover, a random closed surface cuts a sublinear number of geometrical objects associated with the vertices in G in expectation.
- **(IV)** The graph class of G is subgraph-closed.

Compared with HKRS [12], criteria (II)-(IV) replace the minor-closed condition, and can be satisfied by a family of non-minor-closed graph classes. Criteria (I)-(IV) are formally stated in Section 3, in which we use criteria (I)-(IV) to compute a recursive division of the Euclidean graph class in linear time. Once we have the recursive division[1], we can apply HKRS's edge-relaxation algorithm on it to compute SSSP in linear time.

In particular, we show that τ-lanky graphs, the intersection graphs of k-ply neighborhood systems and the intersection graphs of κ-thick cubical neighborhood systems all satisfy the above criteria. Thus we can compute a recursive division (and SSSP) for these Euclidean graph classes in linear time. In this way, we answer Open Problems 2, 3 and 4 affirmatively.

Our approach to constructing a recursive division in linear time uses contracted graphs of the Euclidean graph classes. The main technical challenges of using contracted graphs for Euclidean graph classes are:

- Edge contraction for planar or minor-closed graphs is a combinatorial operation. The contracted graphs in HKRS [12] are obtained by such combinatorial edge contractions. In HKRS's approach, the resulting contracted graphs must remain in the original graph class to guarantee that they have sublinear separators. This is not the case for Euclidean graph classes.
- In Euclidean graph classes, vertices are points in $\mathbb{R}^d$ and edges are segments in $\mathbb{R}^d$. Unfortunately, edge contraction *in the Euclidean space* does not preserve geometric properties and seems meaningless. For example, if an edge $(a, b) \in \mathbb{R}^d$ is contracted into a new vertex $c \in \mathbb{R}^d$, any edge e_a or e_b attached to a or b would now be attached to c. There is no way to choose c so that the geometric properties of e_a and e_b (e.g. length, intersections) are preserved.

Although edge contractions in the Euclidean space are incompatible with the geometric properties of straight line embeddings, we overcome this obstacle by performing edge contractions combinatorially and proving that 1) the combinatorially contracted graphs admit sublinear balanced separators, and 2) such balanced separators can be computed in time linear to the size of the contracted graph. These properties are unknown before and we believe that they are of independent interest.

[1] To be precise, the recursive division should satisfy the conditions in Inequalities (1).

1.2 Techniques

A recursive division of a graph divides the graph into regions, then divides the regions into subregions, and so on recursively. There are two approaches to computing a recursive division of planar graph in linear time. The first approach is to adapt Goodrich's [9] recursive separator decomposition algorithm to compute a recursive division of the planar graph in linear time. Goodrich's recursive separator decomposition algorithm emulates Lipton and Tarjan's planar separator algorithm, maintains tree structures used in LT's algorithm and other tree structures dynamically for the divided pieces during decomposition. In this way a separator of the divided piece can be computed in time sublinear to the size of the piece. The second approach, attributed to Frederickson [7] and HKRS [12], is to contract the graph consecutively into a sequence of contracted graphs, then in reverse order divide and expand the contracted graphs to get a recursive division. The first approach relies on specific properties of planar graphs and maintaining spanning trees dynamically. It seems infeasible to use similar ideas on many other graph classes. The second approach relies on edge contraction and the minor-closed property of planar graphs. However, Euclidean graph classes are not minor-closed, yet edge contraction in the Euclidean space seems meaningless.

We compute a recursive division of the Euclidean graph classes by (1) performing edge contractions on the Euclidean graph combinatorially rather than geometrically and (2) proving that the combinatorially contracted graphs have sublinear separators that can be computed in linear time. To prove that the combinatorially contracted graphs have sublinear separators, for each contracted graph we construct an auxiliary Euclidean graph which we call the *representative graph* of the contracted graph. A representative graph of the contracted graph is a sampled subgraph of the original Euclidean graph. Rather than using the contracted graphs, we use (properties of) their representative graphs to prove that the contracted graphs have sublinear separators.

Moreover, to compute an r'-division of the contracted graph (for different contracted graphs the parameter r' is different), it is normally required that any k'-vertex subgraph of the contracted graph has a sublinear separator, for any k'. This is because when dividing the contracted graphs into smaller and smaller pieces, it is required that the smaller pieces still have sublinear separators. However, the contracted graphs of the Euclidean graph classes do not have this property, even if we use their representative graphs to find the separators. Let the contracted graph be G_i and let the accumulated contraction size be the maximum number of vertices in G that are contracted into a vertex in G_i. We resolve this critical issue by requiring the division size r' to be some constant power of the accumulated contraction size. This requirement forces us to use different parameters for contractions and divisions. This is in contrast to the case in HKRS [12]. Despite this modification, we prove that the effect is sufficient by showing that the induced recursive division is still suitable for applying HKRS's edge-relaxation algorithm in linear time.

1.3 Paper Organization

In Section 2 we give preliminaries that are required for later sections. In Section 3 we prove the main theorem. In Sections 4 we consider SSSP in τ-lanky graphs. Due to space limit, we defer the related work section to the full version of the paper. We defer the section of SSSP in the intersection graph of k-ply neighborhood system and the section of SSSP in the intersection graph of κ-thick cubical neighborhood system to the full version of the paper.

2 Preliminaries

▶ **Definition 5** (r-division). *Given any integer $r > 0$, an r-division of an n-vertex graph divides the graph into $O(n/r)$ regions where each region has $O(r)$ vertices (edges) and $O(\sqrt{r})$ boundary vertices. A region consists of its vertices and edges. A vertex is called a boundary vertex if it is contained in multiple regions.*

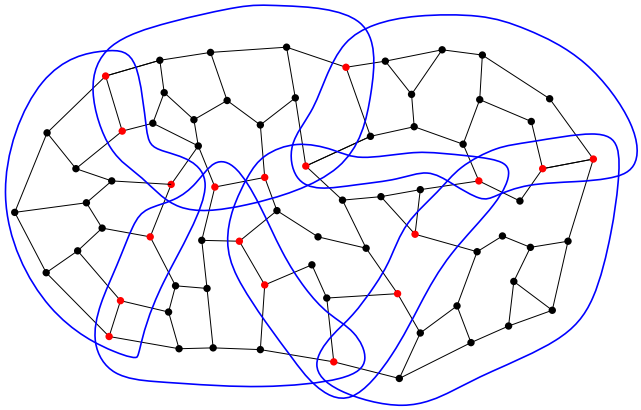


Figure 1 Illustration of r-division. An r-division of the graph divides the graph into regions (enclosed in blue lines) with interior vertices (black) and boundary vertices (red). Modified figure from [7].

▶ **Definition 6** ((r, s)-division). *An (r, s)-division divides an n-vertex graph into $O(n/r)$ regions, such that each region has $r^{O(1)}$ vertices and $O(s)$ boundary vertices.*

A recursive division of a graph divides the graph into regions, then divides the regions into subregions, and so on recursively. Formally, an $(\bar{r}, f)$-recursive division is defined as follows.

▶ **Definition 7** (recursive division [12]). *Let f be a non-decreasing positive integer function and $\bar{r} = (r_0, r_1, \ldots, r_k)$ be a positive integer sequence. An $(\bar{r}, f)$-recursive division of a graph consists of*

- **(i)** *an $(r_k, f(r_k))$-division of the graph, and*
- **(ii)** *an $((r_0, r_1, \ldots, r_{k-1}), f)$-recursive division for each region in the $(r_k, f(r_k))$-division.*

A recursive division can be computed in two phases, the *contraction phase* and the *division phase*. In the contraction phase, a sequence of non-decreasing integers $(z_0, z_1, \ldots, z_I)$ are chosen and a sequence of consecutive contractions are performed. Let $G_0 = G$ and let $\{G_i | 1 \leq i \leq I + 1\}$ be the contracted graphs. Let G_i=Contract(G_{i-1}, z_{i-1}) where the contraction algorithm Contract(G_{i-1}, z_{i-1}) contracts G_{i-1} into G_i. The Contract(G_{i-1}, z_{i-1}) algorithm calls the clustering algorithm in [6] to cluster the vertices of G_{i-1} into $O(n_{i-1}/z_{i-1})$ connected sets (n_{i-1} is the number of vertices in G_{i-1}) with each set containing $O(z_{i-1})$ vertices, then contracts each connected set into a vertex of G_i. The contraction phase ends when the last contracted graph G_{I+1} has at most $n/\log n$ vertices.

▶ **Definition 8.** *Let $G = G_0$ be a graph. We call $\{G_i | 1 \leq i \leq I + 1\}$ the (combinatorially) contracted graphs of G, where $G_i = Contract(G_{i-1}, z_{i-1})$ for some integer z_{i-1}.*

In the division phase, the contracted graphs $G_1, \ldots, G_{I+1}$ are considered in reverse order and a division is obtained for each of the contracted graphs. Let R be an n'-vertex region with a set S of boundary vertices. The Divide(R, S, r) algorithm calls the separator algorithm recursively and computes an (r, s)-division of R (which divides R into subregions). A vertex in a subregion of R is a boundary vertex if it is contained in multiple subregions or it is in S. The division D_{I+1} of G_{I+1} consists of one region, i.e., G_{I+1}. Starting from $i = I$, the division D_i of G_i is obtained from the division D_{i+1} of G_{i+1} by (i) calling Divide(R_{i+1}, S_{i+1}, z_i) for each region R_{i+1} of D_{i+1} and (ii) expanding each vertex of the subregions in R_{i+1} to the connected set in G_i that is contracted into the vertex during the contraction phase. Thus a tree representing the recursive division, called the *recursive division tree*, is obtained.

HKRS's edge-relaxation algorithm. Inspired by Frederickson's work, Henzinger, Klein, Rao and Subramanian [12] developed an optimal $O(n)$ time SSSP algorithm for planar graphs with non-negative edge weights. The algorithm consists of two parts. The first part computes a recursive division of the planar graph in linear time. The second part runs an edge-relaxation algorithm on the recursive division to compute the SSSP. Each node in the recursive division tree represents a region/subregion in the division. Each node (region/subregion) is associated with a heap, whose elements are (the current) distance values to some vertices in the region. The elements in the heap of a region are the min-elements of its subregions' heaps. This forms a hierarchy of heaps. Inside a region, edges are relaxed in analogy to Dijkstra's algorithm, while distance values in the associated heaps are updated. In contrast to Dijkstra's algorithm, the algorithm does not perform edge relaxations for each edge of the region, but only performs a certain number of edge relaxations and then jumps to another region. The numbers of edge relaxations performed inside the regions are carefully coordinated so that (i) when the algorithm terminates, the distance labels of the vertices are the exact distances from the source vertex, (ii) heap operations on large heaps (which are expensive) can be charged to heap operations on small heaps so that the overall heap operations take only $O(n)$ time. In this way the edge-relaxation algorithm correctly computes the distance labels and takes only $O(n)$ time.

A main contribution of HKRS's work is the proof of the following fact. Note that the fact holds for any graph as long as a recursive division satisfying the conditions is given.

▶ **Fact 9** (Section 3 in [12]). *Suppose we are given a graph and an $((r_0, r_1, \ldots, r_k), f)$-recursive division of the graph satisfying*

$$\frac{r_i}{f(r_i)} \geq 8^i f(r_{i-1}) \log r_{i+1} \left(\sum_{j=1}^{i+1} \log r_j \right) \tag{1}$$

for all r_i's exceeding a constant. Then applying the edge-relaxation algorithm on the recursive division takes time linear to the size of the graph.

3 The main theorem

In this section, let G be a Euclidean graph in $\mathbb{R}^d$ that satisfies the following **criteria**:

- **(I)** G admits an $O(c_1 n^{1-1/d})$ balanced separator that can be computed in $O(c_2 n)$ time.
- **(II)** The combinatorially contracted graphs G_i of G are sparse, i.e., any $\bar{k}$-vertex subgraph of G_i has $O(c_3 \bar{k})$ edges.
- **(III)** A closed surface ξ in $\mathbb{R}^d$ (such as a Jordan curve in $\mathbb{R}^2$, a sphere in $\mathbb{R}^d$) is used to find the separator in (I). The closed surface cuts $O(c_4 n^{1-1/d})$ geometrical objects

(such as segments, balls) associated with the vertices in G, and these vertices form the separator. Moreover, a random closed surface cuts $O(c_4 n^{1-1/d})$ geometrical objects (such as segments, balls) associated with the vertices in G in expectation.

(IV) The graph class of G is subgraph-closed.

Here c_1, c_2, c_3 and c_4 are constants. In criteria (I) and (III) we set the exponent of n to be $1 - 1/d$ for ease of exposition. Adapting to any constant < 1 is immediate.

As mentioned in Section 1.2, we will use the edge-relaxation algorithm of HKRS once a recursive division satisfying Inequalities (1) is given. HKRS's edge-relaxation algorithm assumes that the input graph has maximum degree at most 3. Hence, we first transform G into a graph with maximum degree 3. We perform the following transformations on the Euclidean graph G. For each vertex v of degree D greater than 3, let $u_1, \ldots, u_D$ be a cyclic ordering of the vertices adjacent to v. Replace v by a cycle $v_1, \ldots, v_D, v_1$ where each of the vertices v_i $(1 \leq i \leq D)$ is arbitrarily close to v. Replace edge (u_i, v) with new edge (u_i, v_i). Assume that G has $O(c_0 n)$ edges (c_0 is constant), the transformed graph has $O(c_0 n)$ vertices whose maximum degree is 3.

The main task is to compute, in linear time, a recursive division of G that satisfies Inequalities (1). To compute this recursive division, we compute a sequence of combinatorially contracted graphs of G and construct the auxiliary representative graphs of the contracted graphs. In Section 3.1, we use the sparsity of the contracted graphs G_i (criterion (II)) to construct their representative graphs. In Section 3.2 we prove that the (combinatorially) contracted graphs have sublinear separators and such sublinear separators can be found efficiently using their representative graphs. Finally, in Section 3.3, we prove that a recursive division of G satisfying Inequalities (1) can be computed in linear time and conclude the linear time SSSP algorithm.

3.1 The contracted graphs and their representative graphs

Recall that the recursive division can be computed using a two-phase algorithm. In the contraction phase of the recursive division algorithm, the contracted graphs $G_1, G_2, \ldots, G_{I+1}$ are constructed consecutively such that $G_i = Contract(G_{i-1}, z_{i-1})$, $1 \leq i \leq I+1$, where Contract(G_{i-1}, z_{i-1}) is the contraction algorithm. For each contracted graph G_i, $1 \leq i \leq I+1$, we construct its representative graph RG_i.

Each edge of G_i has a *representative edge* in RG_i, which is an edge in G. The representative graph RG_i consists of the representative edges of the edges in G_i and their endpoints. The representative edge of an edge e in G_i is denoted as $rep_{G_i}(e)$. The representative edge of an edge in $G_0 = G$ is the edge itself, and the representative graph of G_0 is G_0.

The representative graph RG_i is constructed from the contracted graph G_{i-1} and its representative graph RG_{i-1}, as follows. Consider a vertex u in G_i and let $cset_{G_{i-1}}(u)$ denote the connected set of vertices in G_{i-1} from which u is contracted. An edge $e = (u, v)$ in G_i corresponds to the set of edges in G_{i-1} between $cset_{G_{i-1}}(u)$ and $cset_{G_{i-1}}(v)$.

Let $REP_{G_i}(e) = \{rep_{G_{i-1}}((u', v')) | u' \in cset_{G_{i-1}}(u), v' \in cset_{G_{i-1}}(v), (u', v') \in E(G_{i-1})\}$ denote the set of the representative edges of the edges in G_{i-1} between $cset_{G_{i-1}}(u)$ and $cset_{G_{i-1}}(v)$. See Figure 2(a) for an illustration. For each edge $e{=}(u, v)$ in $E(G_i)$, choose an arbitrary edge in $REP_{G_i}(e)$ as its representative edge. The chosen representative edges and their endpoints form the representative graph RG_i of G_i.

According to criterion (II), RG_{i-1} has $O(c_3 n_{i-1})$ edges and vertices. It is then observed that RG_i is constructed from G_{i-1} and RG_{i-1} in $O(c_3 n_{i-1})$ time where n_{i-1} is the number of vertices in G_{i-1}. Note that RG_i may not be connected.

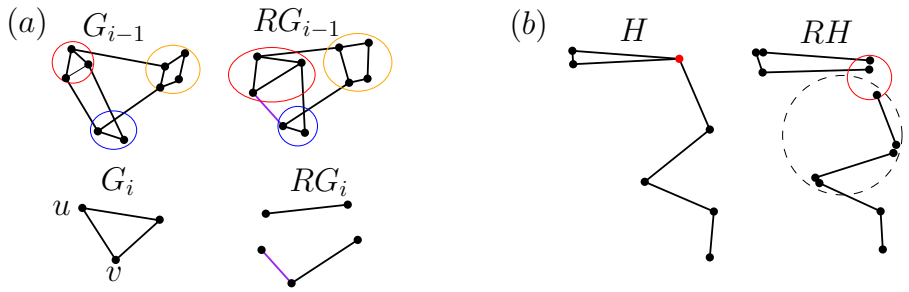


Figure 2 (a) $cset_{G_{i-1}}(u)$ is enclosed in red, $cset_{G_{i-1}}(v)$ is enclosed in blue. The representative edge of (u, v) in G_i is the purple edge. (b) Illustrating the proof of Lemma 11. The closed surface ξ^* is the dashed circle. The red vertex is added to S since its $\Gamma(\cdot)$ has one point inside $\mathbf{b}(o, r^*)$ and two points outside ξ^*.

▶ **Observation 10.** *The representative graph RG_i of G_i can be constructed in $O(c_3 n_{i-1})$ time.*

3.2 Separators of the contracted graphs

The parameters z_i used in the contraction phase, $0 \le i \le I$, are set as in [12]:

$$z_0 = 2, \tag{2}$$

$$z_i = 7^{z_{i-1}^{1/5}}, 1 \le i \le I. \tag{3}$$

Let $\alpha_{i-1} = \prod_{j<i} z_j$ be the accumulated contraction sizes for the contracted graph G_i, and let $\beta_{i-1} = z_{i-1}^{3d-2}$. In [12], parameters z_i $(0 \le i \le I)$ are used for division in the division phase. In contrast, we use β_i in place of z_i as division parameters in the division phase.

A subgraph of G_i with $O(\alpha_{i-1})$ vertices may not admit a sublinear separator. However, the next lemma shows that any subgraph of G_i with $\Omega(\beta_{i-1})$ vertices admit a sublinear separator.

▶ **Lemma 11.** *Let H be any subgraph of G_i with $k = \Omega(\beta_{i-1})$ vertices. A balanced separator of H of size $O((c_1 c_3^{1-1/d} + c_4) k^{(1-\frac{1}{3d-2})})$[2] can be computed in $O(c_2 c_3 k)$ time given H and RH.*

Proof. Let RH denote the representative graph of H, which is a subgraph of RG_i. According to criterion (II), RH has $O(c_3 k)$ vertices. Run the separator algorithm on RH (criterion (I)), noting that RH is a subgraph of G, thus it also admits sublinear separator (criteria (I) and (IV)). Let ξ^* be the closed surface (criterion (III)) used to find the separator of RH. Let S be the desired balanced separator of H, initially being empty. Vertices are added to S in two steps.

(1) For any vertex v in G_i, let $conset_G(v)$ be the (accumulative) connected set of vertices in G that are contracted into v in the contraction phase. For any point p in RG_i, let $rev(p)$ be the vertex v in G_i such that $p \in conset_G(v)$. Note that there is exactly one such vertex v. In the first step, for each geometrical object (criterion (III)) that is cut by ξ^*, add its associated $rev(\cdot)$ to S. For example, if the geometrical object is a ball and p is the ball's center, add $rev(p)$ to S.

[2] We did not try to optimize the size of the separator.

(2) For any vertex v in G_i, let $\Gamma(v)$ be (the set of) endpoints of edges in RG_i that belong to $conset_G(v)$ (we have $\Gamma(v) \subset conset_G(v)$). In the second step, partition the vertices of H into three groups: (i) those such that all points in its $\Gamma(\cdot)$ lie inside ξ^*, (ii) those such that all points in its $\Gamma(\cdot)$ lie outside ξ^*, and (iii) those such that some points in its $\Gamma(\cdot)$ lie inside ξ^* and some points in its $\Gamma(\cdot)$ lie outside ξ^*. Add vertices in group (iii) to S. See Figure 2(b) for an illustration.

We prove that S is a balanced separator of H with $O((c_1c_3^{1-1/d}+c_4)k^{(1-\frac{1}{3d-2})})$ vertices. We first prove the size of S. Since RH is a subgraph of G with $O(c_3k)$ edges and vertices, the closed surface ξ^* cuts $O(c_1(c_3k)^{1-1/d}) = O(c_1c_3^{1-1/d}k^{1-1/d})$ geometrical objects associated with the vertices of RG_i. Since $\Gamma(v)$ is a subset of $conset_G(v)$ and the vertices in $conset_G(v)$ are connected in G, the vertex v belongs to group (iii) only if one or more geometrical objects in (associated with) the subgraph of G induced by vertices in $conset_G(v)$ is cut by ξ^*. Let G_H be the subgraph of G from which H is contracted. G_H has $O(\alpha_{i-1}k)$ vertices where $\alpha_{i-1} = \prod_{j<i} z_j$. Since ξ^* is a random closed surface in G_H, according to criterion (III), ξ^* cuts $O(c_4(\alpha_{i-1}k)^{1-1/d})$ geometrical objects associated with the vertices of G_H in expectation. Since $z_i = 7^{z_{i-1}^{1/5}}$, $\alpha_{i-1} = O(\beta_{i-1}^{\frac{1}{3d-2}}) = O(k^{\frac{1}{3d-2}})$. Thus ξ^* cuts order of

$$c_4(\alpha_{i-1}k)^{1-1/d} \leq c_4k^{(1+\frac{1}{3d-2})(1-1/d)} \tag{4}$$

$$= c_4k^{1-\frac{2d-1}{d(3d-2)}} \leq c_4k^{1-\frac{d}{d(3d-2)}} = c_4k^{1-\frac{1}{3d-2}} \tag{5}$$

geometrical objects associated with G_H. The number of vertices in group (iii) is thus $O(c_4k^{(1-\frac{1}{3d-2})})$. Therefore, the size of S is $O((c_1c_3^{1-1/d}+c_4)k^{(1-\frac{1}{3d-2})})$.

Next, we prove that S is a balanced separator. Let A be the set of vertices in G_i which belong to group (i) and are not added to S. Let B be the set of vertices in G_i which belong to group (ii) and are not added to S. We claim that the removal of S separates A and B. We prove the claim by contradiction.

Each edge in H corresponds to a unique edge in RH. Each vertex v in H corresponds to a set of vertices $\Gamma(v)$ in RH. Let a be a vertex in A and let b be a vertex in B. By the definition of A and B, a is in group (i) so all vertices in $\Gamma(a)$ are inside ξ^* and b is in group (ii) so all vertices in $\Gamma(b)$ are outside ξ^*. If there is an edge in H between a and b, then there is a unique edge (a_r, b_r) in RH such that $a_r \in \Gamma(a)$ and $b_r \in \Gamma(b)$. Since a_r is inside ξ^* and b_r is outside ξ^*, either the geometrical object associated with a_r or the geometrical object associated with b_r would be cut by ξ^*, which implies that either a or b were added to S. A contradiction.

Since ξ^* separates vertices of RH in a balanced way and $|S|$ is sublinear in k, $|A|$ and $|B|$ are proportional to each other. Therefore, S is a balanced separator.

According to criterion (I), running the separator algorithm on RH takes $O(c_2c_3k)$ time. This finishes the proof of the lemma. ◀

3.3 Recursive division

The recursive division of G is computed in two phases. In the contraction phase, the contraction parameters z_i are set as Equations (2) and (3), and the contracted graphs $G_1, \ldots, G_{I+1}$ are computed consecutively by the Contract() algorithm. In the division phase, we choose the parameters for dividing G_i, $1 \leq i \leq I+1$, to be $\beta_{i-1} = z_{i-1}^{3d-2}$. Since the minimum subgraph of G_i that will be divided has $\Omega(\beta_{i-1})$ vertices, we can apply Lemma 11 recursively to obtain the division D_i of G_i.

To analyze the computing time of the recursive division, we analyze the division D_i of G_i, and the division of G it induces. Recall that the division D_i is derived from the division D_{i+1} of G_{i+1}. Let R_{i+1} be a region in D_{i+1}. The Divide$(R_{i+1}, S_{i+1}, \beta_i)$ algorithm divides R_{i+1} into subregions each of which has at most $\beta_i = z_i^{3d-2}$ vertices and has $O((c_1 c_3^{1-1/d} + c_4) z_i^{3d-3})$ boundary vertices. Each expanded subregion, which is a region in D_i, has at most $\beta_i \cdot 3z_i = 3z_i^{3d-1}$ vertices and $O((c_1 c_3^{1-1/d} + c_4) z_i^{3d-2})$ boundary vertices. Let k_i denote the number of regions in the division D_i, $0 \le i \le I+1$, and let n_i be the number of vertices in G_i. We can prove, using similar arguments as in the proof of Lemma 4.1 in [12] that

▶ **Lemma 12.** *The number of regions in the division D_i is $O(n_i / z_i^{3d-1})$.*

From the proof of Lemma 12, we can also prove that the sum of the vertices of all regions in D_i is $O(n_i)$. Now we can analyze the computing time of the recursive division of G.

▶ **Lemma 13.** *Let G be an Euclidean graph in $\mathbb{R}^d$ that satisfies the criteria (I)-(IV) and has $O(c_0 n)$ edges. A recursive division of G can be computed in $O(c_0 c_2 c_3 dn)$ time.*

Proof. We calculate the costs in steps.

1. The number of edges in G_i, $i \ge 1$, is $O(c_3 n_i)$ by criterion (II). Thus, computing the contracted graphs $G_1, \ldots, G_{I+1}$ takes time $O(c_0 n + c_3 n_1 + \ldots + c_3 n_I)$, which is $O(c_0 c_3 n)$.
2. By Observation 10, computing the representative graphs $RG_1, \ldots, RG_{I+1}$ takes $O(c_0 n + c_3 n_1 + \ldots + c_3 n_I) = O(c_0 c_3 n)$ time.
3. For $i \le I$, divide a region R_{i+1} in the division D_{i+1} of G_{i+1} takes $O(c_2 c_3 |R_{i+1}| \log |R_{i+1}|)$ time, by recursively applying Lemma 11.
4. Each region R_{i+1} in D_{i+1} has $O(z_{i+1}^{3d-1})$ vertices. Summing over all regions in D_{i+1}, dividing D_{i+1} takes $O(c_2 c_3 d \sum_{R_{i+1}} |R_{i+1}| \log z_{i+1})$ time, which is $O(c_2 c_3 d n_{i+1} \log z_{i+1})$ by the fact that $\sum_{R_{i+1}} |R_{i+1}|$ is $O(n_{i+1})$.
5. The time to obtain the division D_i of G_i after dividing D_{i+1} is $O(c_3 n_i)$. Computing all the divisions D_i, $1 \le i \le I+1$, thus takes $O(c_2 c_3 d \sum_i n_i \log z_i)$ time. Since $n_i \le n_{i-1}/z_{i-1} \le c_0 n / z_{i-1}$ and $\log z_i$ is $O(z_{i-1}^{1/5})$, $O(c_2 c_3 d \sum_i n_i \log z_i)$ is $O(c_2 c_3 d c_0 n) = O(c_0 c_2 c_3 dn)$.

   Therefore, the total running time is $O(c_0 c_2 c_3 dn)$, which finishes the proof of the lemma. ◀

Finally, we prove that the recursive division satisfies Inequalities (1), which will imply that applying HKRS's edge-relaxation algorithm on it will take $O(c_0 n)$ time.

▶ **Lemma 14.** *The recursive division of G satisfies Inequalities* (1).

Proof. Since $n_j \le n_{j-1}/z_{j-1}$, we have $n_i \le c_0 n / \prod_{j<i} z_j$. Note also that each vertex in G_i expands to at most $\prod_{j<i} 3z_j$ vertices of G.

Consider the division D_i of G_i, and the division of G it induces. The division D_i consists of $O(n_i / z_i^{3d-1})$ regions (Lemma 12), each of which has $O(z_i^{3d-1})$ vertices and $O((c_1 c_3^{1-1/d} + c_4) z_i^{3d-2})$ boundary vertices. Thus it induces a division of G consisting of $O(n_i / z_i^{3d-1})$ regions, each of which has $O(z_i^{3d-1} \prod_{j<i} 3z_j)$ vertices and $O((c_1 c_3^{1-1/d} + c_4) z_i^{3d-2} \prod_{j<i} 3z_j)$ boundary vertices.

Let $r_i = z_i^{3d-1} \prod_{j<i} z_j$, and let

$$f(r_i) = c'(c_1 c_3^{1-1/d} + c_4) z_i^{3d-2} \prod_{j<i} z_j.$$

The induced division of G has $O(c_0 n / r_i)$ regions each having $O(r_i 3^i)$ vertices and $O(f(r_i))$ boundary vertices. Since $3^i \le \prod_{j \le i} z_j$, each region has $O(r_i^2)$ vertices.

We have

$$\frac{r_i}{f(r_i)} = \frac{z_i^{3d-1}}{c'(c_1 c_3^{1-1/d} + c_4) z_i^{3d-2}} = \frac{z_i}{c'(c_1 c_3^{1-1/d} + c_4)}.$$

By the setting of z_i, we have $z_{i-1} = \Theta(\log^5 z_i)$ and $\prod_{j<i} z_j = O(\log^6 z_i)$. Therefore

$$f(r_{i-1}) = c'(c_1 c_3^{1-1/d} + c_4) z_{i-1}^{3d-2} \prod_{j<i-1} z_j = O((c_1 c_3^{1-1/d} + c_4) \log^{5(3d-2)} z_i \log^6 \log z_i).$$

We have $\log r_{i+1} = O(\log z_{i+1}^{3d-1}) = O(d \cdot z_i^{1/5})$ and $\sum_{j=1}^{i+1} \log r_j = O(d \cdot z_i^{1/5})$. Since c_1, c_3, c_4, d are all constants, for a sufficiently large constant $\hat{c}$, Inequalities (1) hold for all r_i exceeding $\hat{c}$. ◀

We have obtained the main theorem of this section.

▶ **Theorem 15.** *Let G be an Euclidean graph in $\mathbb{R}^d$ (d is fixed) that satisfies criteria (I)-(IV) and has $O(c_0 n)$ edges. Then a $(\bar{r}, f)$-recursive division of G satisfying Inequalities* (1) *can be computed in $O(c_0 c_2 c_3 dn)$ time.*

As a corollary,

▶ **Corollary 16.** *Let G be an Euclidean graph in $\mathbb{R}^d$ (d is fixed) that satisfies criteria (I)-(IV) and has $O(c_0 n)$ edges. Single-source shortest path in G can be computed in $O(c_0 c_2 c_3 dn)$ time.*

Note that since edge-relaxation is comparison-based and does not make any assumptions about edge weights, the edge weights of the Euclidean graph can be any nonnegative value, not just the Euclidean distance between the endpoints.

4 SSSP for lanky-graphs

Lanky graphs were introduced by Le and Than [15] when studying sublinear separators of greedy spanners in Euclidean and doubling metrics.

▶ **Definition 17.** *A graph G embedded in an Euclidean or a doubling metric is τ-lanky, if for any $r > 0$ and any ball of radius r, there are at most τ edges of length at least r that are cut by the ball. An edge is cut by a ball if one endpoint of the edge is inside the ball and the other endpoint of the edge is outside the ball.*

A number of graphs are $O(1)$-lanky graphs, including greedy spanners in Euclidean and doubling metrics, low-density graphs in Euclidean [1] and doubling metrics, greedy spanners for point sets in $\mathbb{R}^d$ of low fractal dimension and greedy spanners for unit ball graphs in $\mathbb{R}^d$.

To unify the construction of sublinear separators for lanky graphs in both Euclidean and doubling metrics, Le and Than proposed the (η, d)-packable metric space. We say that a set of points P is r-separated if the distance between any two points in P is at least r.

▶ **Definition 18.** *A* metric (X, δ_X) *is (η, d)-packable if for any $r \in (0, 1]$ and any r-separated set $P \subseteq X$ contained in a unit ball, $|P| \leq \eta(\frac{1}{r})^d$. We call d the packing dimension of the* metric *and η the packing constant of the* metric.

The authors proved the following separator theorem for lanky graphs in an (η, d)-packable metric space.

▶ **Theorem 19** ([15]). *Let (X, δ_X) be an (η, d)-packable metric space and let $G = (V, E, w)$ be a τ-lanky graph in (X, δ_X). G has a $(1 - \frac{1}{\eta 2^{d+1}})$-balanced separator S such that S has size $O(\tau\eta 8^d n^{1-1/d})$. S can be computed in $O((\eta^3 8^d + \tau)n)$ expected time.*

The separator algorithm in the proof of Theorem 19 consists of two steps.

1. Compute a ball $\mathbf{b}(v, r)$ such that $\mathbf{b}(v, r)$ contains at least $\frac{n}{\eta 2^{d+1}}$ vertices of G and the concentric ball $\mathbf{b}(v, 2r)$ contains at most $\frac{n}{2}$ vertices of G, using the randomized algorithm in [11].
2. Choose $r^* = (1 + \sigma)r$ where $\sigma \in (0, 1]$ is chosen uniformly at random. Let $\mathbf{b}(v, r^*)$ be the random ball. Compute the set of edges E^* of G that are cut by $\mathbf{b}(v, r^*)$. Return the endpoints of the edges in E^* as the separator.

With constant probability, E^* has size $O(\tau\eta 8^d n^{1-1/d})$. This is proved by partitioning the edges in E^* into groups of exponentially increasing lengths and summing over the groups. From the proof of Theorem 19, it can be inferred that a random ball $\mathbf{b}(x, y)$ cuts $O(\tau\eta 8^d n^{1-1/d})$ edges of G in expectation.

Until now, we have verified that a τ-lanky graph in an (η, d)-packable metric space satisfies criteria (I) and (III) in Section 3 with $c_1 = \tau\eta 8^d$, $c_2 = \eta^3 8^d + \tau$ and $c_4 = \tau\eta 8^d$. The closed surface ξ used to find the separator is a ball. The geometrical objects cut by ξ are segments, which are edges in G.

From the definition of lanky graphs, it is easy to verify they are subgraph-closed. Thus criterion (IV) is satisfied. It remains to verify criterion (II).

4.1 Sparsity of the contracted graphs

We relate the sparsity of the combinatorially contracted graphs of a lanky graph to the graph-theoretic property *thickness*. The thickness of a graph G' is the minimum number of planar subgraphs that G' can be decomposed into. Let $\theta(G')$ denote the thickness of a graph G'. Determining the thickness of a general graph is NP-hard [17]. However, for graphs with maximum degree Δ, Halton [10] proved that if G' is a graph with maximum degree Δ then $\theta(G') \leq \lceil \Delta/2 \rceil$. It follows from the definition of lanky graph that a τ-lanky graph G has maximum degree τ. Therefore the thickness of G is at most $\lceil \tau/2 \rceil$.

Thus, the τ-lanky graph G can be decomposed into $\lceil \tau/2 \rceil$ planar subgraphs, which we call $H_1, \ldots, H_{\lceil \tau/2 \rceil}$. Let each of $H_1, \ldots, H_{\lceil \tau/2 \rceil}$ have $V(G)$ as its vertex set. Use the clustering inside the Contract(G, z_0) algorithm to cluster vertices in $V(G)$. For each H_j, $1 \leq j \leq \lceil \tau/2 \rceil$, shrink each cluster into a vertex while retaining edges between the clusters (shrunk vertices). Let Shrink(H_j, z_0) be the shrunk graph of H_j. Then Contract(G, z_0) is the union of Shrink(H_j, z_0), $1 \leq j \leq \lceil \tau/2 \rceil$ (removing multi-edges).

Since Shrink(H_j, z_0) is a planar graph, $G_1 = Contract(G, z_0)$ has $O(\tau n_1)$ edges where n_1 is the number of vertices in G_1. Similarly G_i, $i \geq 2$, is the union of the iteratively shrunk graphs of H_j, $1 \leq j \leq \lceil \tau/2 \rceil$, thus has $O(\tau n_i)$ edges where n_i is the number of vertices in G_i.

▶ **Lemma 20.** *The contracted graph G_i of G, $1 \leq i \leq I + 1$, has $O(\tau n_i)$ edges.*

Following similar ideas, we get:

▶ **Corollary 21.** *If H is a subgraph of G_i with $\bar{k}$ vertices then H has $O(\tau\bar{k})$ edges.*

This justifies that a τ-lanky graph satisfies criterion (II) with $c_3 = \tau$.

Since G has maximum degree τ, G has $O(\tau n)$ edges so $c_0 = \tau$. Plugging the constants into Theorem 15, we have

▶ **Theorem 22.** *Let G be a τ-lanky graph in an (η, d)-packable metric space. A $(\bar{r}, f)$-recursive division of G satisfying Inequalities* (1) *can be computed in $O((\tau^2\eta^3 8^d + \tau^3)dn)$ randomized time.*

As a corollary, we get

▶ **Corollary 23.** *Let G be a τ-lanky graph in an (η, d)-packable metric space. Single-source shortest path in G can be computed in $O((\tau^2\eta^3 8^d + \tau^3)dn)$ time.*

References

1 Daniel Chen, Anne Driemel, Leonidas J. Guibas, Andy Nguyen, and Carola Wenk. Approximate map matching with respect to the Fréchet distance. In *Proceedings of the Thirteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2011, Holiday Inn San Francisco Golden Gateway, San Francisco, California, USA, January 22, 2011*, pages 75–83. SIAM, 2011. doi:10.1137/1.9781611972917.8.

2 Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. doi:10.1007/BF01386390.

3 David Eppstein and Michael T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2008*, page 16. ACM, 2008. doi:10.1145/1463434.1463455.

4 David Eppstein and Hadi Khodabandeh. On the edge crossings of the greedy spanner. In *37th International Symposium on Computational Geometry, SoCG 2021*, volume 189 of *LIPIcs*, pages 33:1–33:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.SOCG.2021.33.

5 David Eppstein, Gary L. Miller, and Shang-Hua Teng. A deterministic linear time algorithm for geometric separators and its applications. In *Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 99–108. ACM, 1993. doi:10.1145/160985.161005.

6 Greg N. Frederickson. Data structures for on-line updating of minimum spanning trees, with applications. *SIAM J. Comput.*, 14(4):781–798, 1985. doi:10.1137/0214055.

7 Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987. doi:10.1137/0216064.

8 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 338–346. IEEE Computer Society, 1984. doi:10.1109/SFCS.1984.715934.

9 Michael T. Goodrich. Planar separators and parallel polygon triangulation. *J. Comput. Syst. Sci.*, 51(3):374–389, 1995. doi:10.1006/JCSS.1995.1076.

10 John H. Halton. On the thickness of graphs of given degree. *Inf. Sci.*, 54(3):219–238, 1991. doi:10.1016/0020-0255(91)90052-V.

11 Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006. doi:10.1137/S0097539704446281.

12 Monika Rauch Henzinger, Philip N. Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.*, 55(1):3–23, 1997. doi:10.1006/JCSS.1997.1493.

13 Casimir Kuratowski. Sur le probleme des courbes gauches en topologie. *Fundamenta mathematicae*, 15(1):271–283, 1930.

14 Hung Le and Cuong Than. Greedy spanners in Euclidean spaces admit sublinear separators. *CoRR*, abs/2107.06490, 2021. arXiv:2107.06490.

15 Hung Le and Cuong Than. Greedy spanners in Euclidean spaces admit sublinear separators. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 3287–3310. SIAM, 2022. doi:10.1137/1.9781611977073.130.

16 R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math*, 36(2):177–189, 1979.

17 Anthony Mansfield. Determining the thickness of graphs is np-hard. *Math. Proc. Cambridge Philos. Soc.*, 9:9–23, 1983.

18 Gary L. Miller, Shang-Hua Teng, William P. Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, 1997. doi:10.1145/256292.256294.

19 Gary L. Miller, Shang-Hua Teng, William P. Thurston, and Stephen A. Vavasis. Geometric separators for finite-element meshes. *SIAM J. Sci. Comput.*, 19(2):364–386, 1998.

20 Gary L. Miller, Shang-Hua Teng, William P. Thurston, and Stephen A. Vavasis. Geometric separators for finite-element meshes. *SIAM J. Sci. Comput.*, 19(2):364–386, 1998.

21 Gary L. Miller, Shang-Hua Teng, and Stephen A. Vavasis. A unified geometric approach to graph separators. In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 538–547. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185417.

22 Warren D. Smith and Nicholas C. Wormald. Geometric separator theorems & applications. In *39th Annual Symposium on Foundations of Computer Science, FOCS 1998*, pages 232–243. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743449.

23 Siamak Tazari and Matthias Müller-Hannemann. A faster shortest-paths algorithm for minor-closed graph classes. In *Graph-Theoretic Concepts in Computer Science, 34th International Workshop, WG 2008*, volume 5344 of *Lecture Notes in Computer Science*, pages 360–371, 2008. doi:10.1007/978-3-540-92248-3_32.

Optimal Randomized Clustering of Matrices

Mustafa Alper Gunes
Department of Mathematics, Princeton University, NJ, USA

Assaf Naor
Department of Mathematics, Princeton University, NJ, USA

Abstract

If $\mathbf{X} = (\mathsf{M}_n(\mathbb{R}), \|\cdot\|_{\mathbf{X}})$ is a unitarily invariant normed space, i.e., $\|\mathsf{UAV}\|_{\mathbf{X}} = \|\mathsf{A}\|_{\mathbf{X}}$ for every matrix $\mathsf{A} \in \mathsf{M}_n(\mathbb{R})$ and every two orthogonal matrices $\mathsf{U}, \mathsf{V} \in \mathsf{M}_n(\mathbb{R})$, then we evaluate up to universal constant factors the smallest $\sigma > 0$ for which there is a probability distribution over partitions of $\mathbf{X}$ into clusters of diameter at most 1 yet for every two matrices $\mathsf{A}, \mathsf{B} \in \mathsf{M}_n(\mathbb{R})$ the probability that they fall into distinct clusters is at most σ times the $\mathbf{X}$-distance between A and B. Specifically, we prove that this infimal σ, which is called the separation modulus of $\mathbf{X}$ and is denoted $\mathsf{SEP}(\mathbf{X})$, satisfies:

$$\mathsf{SEP}(\mathbf{X}) = \Theta\big(\sqrt{n} \cdot \|\mathsf{I}_n\|_{\mathbf{X}} \cdot \mathrm{diam}(B_{\mathbf{X}})\big), \tag{1}$$

where I_n is the n-by-n identity matrix and $\mathrm{diam}(B_{\mathbf{X}})$ is the diameter with respect to the standard Euclidean metric on $\mathsf{M}_n(\mathbb{R})$ of the unit ball $B_{\mathbf{X}}$ of $\mathbf{X}$. Our proof of (1) proceeds through an asymptotic evaluation of the spectral gap of the Laplacian with Dirichlet boundary conditions on $B_{\mathbf{X}}$, which we achieve by exact computations for a Jacobi orthogonal random matrix ensemble. Assuming oracle access to norm evaluations in $\mathbf{X}$, by combining (1) with a new deterministic algorithm for a $O(1)$-approximation of the diameter of convex bodies in $\mathbb{R}^n$ that are given by a weak membership oracle and are symmetric with respect to coordinate permutations and reflections about the standard axes (this task is famously known to be impossible in the absence of such symmetries), we get an oracle polynomial time algorithm whose output is the separation modulus of $\mathbf{X}$ up to universal constant factors. Another example of a consequence of (1) is that for each $m \in \{1, \ldots, n\}$ the separation modulus of the m'th Ky Fan norm on $\mathsf{M}_n(\mathbb{R})$ is bounded from above and from below by universal constant multiples of $m\sqrt{n}$ if $m \geqslant \sqrt{n}$, and of n if $m \leqslant \sqrt{n}$. We also deduce from (1) an upper bound on the Lipschitz extension modulus of $\mathbf{X}$ that improves over the previously best-known bound even in the special case when $\mathbf{X}$ is $\mathsf{M}_n(\mathbb{R})$ equipped with the $\ell_2^n \to \ell_2^n$ operator norm.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Mathematical analysis; Theory of computation → Design and analysis of algorithms

Keywords and phrases Clustering, Unitarily Invariant Matrix Norms, Oracle Polynomial Time Approximation Algorithms for Radii of Convex Bodies, Extension of Lipschitz Functions, Random Matrices, Spectrum of the Laplacian with Dirichlet Boundary Conditions, Reverse Isoperimetry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.56

Related Version *Full Version*: `https://arxiv.org/pdf/2508.03853.pdf`

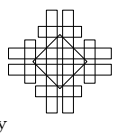
Funding *Assaf Naor*: supported by NSF grant DMS-2453936, BSF grant 2018223, and a Simons Investigator award.

1 Introduction

For $\sigma > 0$, a metric space $(\mathcal{M}, d_{\mathcal{M}})$ is said to admit a σ-separating random partition if for every $\Delta > 0$ there is a distribution over random partitions of $\mathcal{M}$ into clusters of diameter at most Δ with the property that for *every* two points $x, y \in \mathcal{M}$ the probability that x and y belong to distinct clusters is at most σ/Δ times the distance between x and y. The smallest σ for which this is possible is called the separation modulus of $\mathcal{M}$ and it is denoted $\mathsf{SEP}(\mathcal{M})$.


© Mustafa Alper Gunes and Assaf Naor;
licensed under Creative Commons License CC-BY 4.0





Separating random partitions are a well studied clustering primitive which was introduced and investigated in [10] (precursors that considered it implicitly, with a variety of applications, appear in [49, 7, 1, 40]); see e.g. [54, 55] for more on the history. By [10], every n-point metric space $(\mathcal{M}, d_{\mathcal{M}})$ satisfies $\mathsf{SEP}(\mathcal{M}) = O(\log n)$, and there are metric spaces for which this bound is sharp. In [45], a randomized polynomial time algorithm was designed that takes as input a finite metric space $(\mathcal{M}, d_{\mathcal{M}})$ and outputs a factor 2 approximation to $\mathsf{SEP}(\mathcal{M})$. See [11, 20, 41, 31, 46, 24, 47, 45, 8, 54, 55, 43, 56, 44, 57] for an indication of the literature on this topic. The investigation of separating random partitions of finite-dimensional normed spaces, which is what we study herein, originates in [58], motivated by applications to network routing and distributed computing. The work [20] sharpened and generalized the bounds of [58], and influenced later research, e.g. [47, 4, 55]; similar partitioning schemes appeared implicitly in the earlier work [37] on SDP-based algorithms for graph colorings. Applications of separating random partitions to pure mathematics have also been found, notably to metric embeddings [11, 24, 56] and extension of vector-valued Lipschitz functions [47, 55].

In Theorem 1 below we obtain an asymptotic formula (up to universal constant factors) for the separation modulus of $\mathsf{M}_n(\mathbb{R})$ equipped with an arbitrary unitarily invariant matrix norm;[1] we will spell out all of its (standard) notation and terminology right after stating it:

▶ **Theorem 1.** *Let* $\mathbf{X} = (\mathsf{M}_n(\mathbb{R}), \|\cdot\|_{\mathbf{X}})$ *be a unitarily invariant normed space. Then*[2]

$$\mathsf{SEP}(\mathbf{X}) \asymp \sqrt{n}\|\mathsf{I}_n\|_{\mathbf{X}} \operatorname{diam}(B_{\mathbf{X}}), \tag{2}$$

where $\operatorname{diam}(B_{\mathbf{X}}) = \operatorname{diam}_{\mathsf{S}_2^n}(B_{\mathbf{X}})$ *is the diameter with respect to the standard Euclidean metric on* $\mathsf{M}_n(\mathbb{R})$ *of the unit ball* $B_{\mathbf{X}}$ *of* $\mathbf{X}$*, and* $\mathsf{I}_n \in \mathsf{M}_n(\mathbb{R})$ *is the* n*-by-*n *identity matrix.*

A norm $\|\cdot\|$ on the space $\mathsf{M}_n(\mathbb{R})$ of n-by-n matrices with real entries is said to be unitarily invariant if $\|\mathsf{UAV}\| = \|\mathsf{A}\|$ for every $\mathsf{A} \in \mathsf{M}_n(\mathbb{R})$ and every two matrices U, V that belong to the orthogonal group O_n. Norms with this property [71, 64] form a rich class of ways to measure the size of matrices in an intrinsic manner that does not depend on the choice of orthogonal bases of $\mathbb{R}^n$ with respect to which the matrix representation of the corresponding linear operator is formed. As such, unitarily invariant norms are of great importance and utility to a wide range of disciplines, ranging from multiple areas of pure mathematics and theoretical physics, to algorithm design, signal processing, quantum information theory, numerical linear algebra, shape analysis and imaging, statistics, machine learning and data

[1] Formally, when dealing with infinite metric spaces (in particular, in the case of unitarily invariant matrix norms that we study herein), one must impose some measurability requirement for the definition of the separation modulus to make sense. At the very least, one has to demand that for every $x, y \in \mathcal{M}$ the aforementioned event "x and y belong to distinct clusters" is measurable with respect to the underlying probability measure on the space of partitions of $\mathcal{M}$. For the purpose of the present article, it suffices to assume this "minimal" measurability. However, it is beneficial for applications to impose stronger measurability assumptions; such a foundational treatment of random partitions of infinite spaces has been carried out in [47, 55]. One of the applications of Theorem 1 is to deduce an improved extension result for Lipschitz functions (Theorem 2 below), for which the aforementioned stronger measurability assumptions are needed. However, this will be achieved by substituting the ensuing reasoning into the framework of [55], which is shown in [55] to indeed yield the desired measurability. Thus, we do not need to recall stronger measurability requirements as they do not directly pertain to the ensuing discussion. Alternatively, our results are meaningful and new even if one only treats random partitions of finite subsets of $\mathsf{M}_n(\mathbb{R})$ equipped with a unitarily invariant matrix norm rather than random partitions of all of $\mathsf{M}_n(\mathbb{R})$, a setting to which measurability considerations are irrelevant.

[2] In addition to the usual $O(\cdot), o(\cdot), \Omega(\cdot), \Theta(\cdot)$ notation, we will use throughout this text the following conventions for asymptotic notation: Given $a, b > 0$, by writing $a \lesssim b$ or $b \gtrsim a$ we mean that $a \leqslant \kappa b$ for some universal constant $\kappa > 0$, and $a \asymp b$ stands for $(a \lesssim b) \wedge (b \lesssim a)$; when we will need to allow for dependence on parameters, we will indicate it by subscripts.

analysis; we point to this selection [73, 50, 75, 13, 67, 62, 32, 28, 22, 59, 70, 69, 72, 63, 68] of sources for part of the massive and varied literature on this (examining the references therein and those that arise from internet search leads to a cornucopia of additional relevant works).

Non-examples of unitarily invariant norms on matrices include those that consider a matrix $\mathsf{A} = (a_{ij}) \in \mathsf{M}_n(\mathbb{R})$ to merely be a collection of n^2 numbers, e.g., the $\ell_p^{n^2}$ norms

$$\|\mathsf{A}\|_{\ell_p^{n^2}} \stackrel{\text{def}}{=} \Big(\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^p\Big)^{\frac{1}{p}} \tag{3}$$

for $p \in [1,\infty] \setminus \{2\}$; such measurements of the size of a matrix are of course highly valuable even though they ignore the operator meaning of the arrangement of those numbers as an n-by-n array, and they are *not* what we study herein (for the aforementioned $\ell_p^{n^2}$ norm, the separation modulus has been evaluated up to universal constant factors in [20] when $1 \leqslant p \leqslant 2$ and in [55] when $p > 2$). Examples of unitarily invariant norms include the Schatten von-Neumann trace class S_p^n for every $1 \leqslant p \leqslant \infty$, which are given by setting

$$\forall \mathsf{A} \in \mathsf{M}_n(\mathbb{R}), \qquad \|A\|_{\mathsf{S}_p^n} \stackrel{\text{def}}{=} \Big(\operatorname{Trace}\big((\mathsf{A}\mathsf{A}^*)^{\frac{p}{2}}\big)\Big)^{\frac{1}{p}} = \Big(\sum_{i=1}^n s_i(\mathsf{A})^p\Big)^{\frac{1}{p}},$$

where $s_1(\mathsf{A}) \geqslant \ldots \geqslant s_n(\mathsf{A}) \geqslant 0$ are the singular values of A, i.e., they are the (decreasing rearrangement of the) eigenvalues of the positive semidefinite matrix $|\mathsf{A}| = (\mathsf{A}\mathsf{A}^*)^{1/2}$. If $p = \infty$, then S_∞^n coincides with the space of n-by-n matrices equipped with their operator (a.k.a. spectral) norm, when they are viewed as linear operators from ℓ_2^n to itself, i.e.,

$$\|\mathsf{A}\|_{\mathsf{S}_\infty^n} = \max_{x \in B_{\ell_2^n}} \|\mathsf{A}x\|_{\ell_2^n} = s_1(\mathsf{A}),$$

where $B_{\mathbf{X}} = \{x \in \mathbf{X} : \ \|x\|_{\mathbf{X}} \leqslant 1\}$ denotes the unit ball of a normed space $(\mathbf{X}, \|\cdot\|_{\mathbf{X}})$. Often S_1^n is called the nuclear norm on $\mathsf{M}_n(\mathbb{R})$, and S_2^n is the standard Euclidean (a.k.a. Hilbert–Schmidt or Frobenius) norm on $\mathsf{M}_n(\mathbb{R})$, i.e., it coincides with the case $p = 2$ of (3). Additional examples are furnished [25] by the Ky Fan norms $\|\cdot\|_{\mathsf{K}_1}, \ldots, \|\cdot\|_{\mathsf{K}_n}$ that are defined by:

$$\forall m \in [n] \stackrel{\text{def}}{=} \{1,\ldots,n\}, \ \forall \mathsf{A} \in \mathsf{M}_n(\mathbb{R}), \qquad \|A\|_{\mathsf{K}_m^n} \stackrel{\text{def}}{=} \sum_{i=1}^m s_i(\mathsf{A}). \tag{4}$$

More generally, let $\mathbf{E} = (\mathbb{R}^n, \|\cdot\|_{\mathbf{E}})$ be a symmetric normed space, i.e., the following requirement holds for every vector $x = (x_1,\ldots,x_n) \in \mathbb{R}^n$, every choice of permutation $\pi \in S_n$, and every choice of signs $\varepsilon_1,\ldots,\varepsilon_n \in \{-1,1\}$:

$$\|(\varepsilon_1 x_{\pi(1)},\ldots,\varepsilon_n x_{\pi(n)})\|_{\mathbf{E}} = \|x\|_{\mathbf{E}}. \tag{5}$$

The unitary ideal $\mathsf{S}_{\mathbf{E}}^n$ of $\mathbf{E}$ is defined to be $\mathsf{M}_n(\mathbb{R})$ equipped with the norm that is given by:

$$\forall \mathsf{A} \in \mathsf{M}_n(\mathbb{R}), \qquad \|\mathsf{A}\|_{\mathsf{S}_{\mathbf{E}}^n} = \big\|\big(s_1(\mathsf{A}),\ldots,s_n(\mathsf{A})\big)\big\|_{\mathbf{E}}. \tag{6}$$

A norm $\|\cdot\|$ on $\mathsf{M}_n(\mathbb{R})$ is unitarily invariant iff there is a symmetric norm $\|\cdot\|_{\mathbf{E}}$ on $\mathbb{R}^n$ such that $\|\cdot\| = \|\cdot\|_{\mathsf{S}_{\mathbf{E}}^n}$; see e.g. [53] or [12, Chapter IV] for a justification of this classical fact.

The novel content of Theorem 1 is the upper bound on $\mathsf{SEP}(\mathbf{X})$ in (2), i.e., Theorem 1 provides an improved randomized clustering of $\mathbf{X}$. The matching impossibility result is due to [55, Corollary 79], where it was proved that $\sqrt{n}\|\mathsf{I}_n\|_{\mathbf{X}} \operatorname{diam}(B_{\mathbf{X}}) \lesssim \mathsf{SEP}(\mathbf{X}) \lesssim$

$\sqrt{n\log n}\|\mathsf{I}_n\|_{\mathbf{X}}\operatorname{diam}(B_{\mathbf{X}})$ in the setting of Theorem 1. Therefore, what Theorem 1 achieves is removing the last redundant unbounded lower-order factor, thus obtaining an asymptotic evaluation of $\mathsf{SEP}(\mathbf{X})$ that is optimal up universal constant factors.

For the operator norm, the aforementioned bounds from [55] are $n \lesssim \mathsf{SEP}(\mathsf{S}_\infty^n) \lesssim n\sqrt{\log n}$, and these were the previously best-known estimates on $\mathsf{SEP}(\mathsf{S}_\infty^n)$. By Theorem 1, we now know that $\mathsf{SEP}(\mathsf{S}_\infty^n) \asymp n$.

As another example, it is straightforward to check from (4) that for each $m \in [n]$ the S_2^n-diameter of the unit ball of the m'th Ky Fan norm K_m^n equals $2\sqrt{n+m(m-1)}/m \asymp \max\{1, \sqrt{n}/m\}$, so Theorem 1 demonstrates that $\mathsf{SEP}(\mathsf{K}_m^n)$ is bounded from above and from below by positive universal constant multiples of $m\sqrt{n}$ if $m \geqslant \sqrt{n}$, and of n if $m \leqslant \sqrt{n}$, thus providing an interpolation between the operator norm $\mathsf{K}_1^n = \mathsf{S}_\infty^n$ and the nuclear norm $\mathsf{K}_n^n = \mathsf{S}_1^n$ that somewhat curiously stabilizes (up to universal constant factors) at $m \asymp \sqrt{n}$.

Theorem 1 implies progress on the important and extensively studied classical topic of extending vector-valued Lipschitz functions; see e.g. [19, 55] and the references therein. The Lipschitz extension modulus of a metric space $(\mathcal{M}, d_{\mathcal{M}})$, denoted $\mathsf{e}(\mathcal{M}, d_{\mathcal{M}})$ or simply $\mathsf{e}(\mathcal{M})$ when the underlying metric $d_{\mathcal{M}}$ is clear from the context, is the infimum over those $L \in [1,\infty]$ such that for every subset $\mathcal{C} \subseteq \mathcal{M}$, every Banach space $(\mathbf{Y}, \|\cdot\|_{\mathbf{Y}})$, and every 1-Lipschitz function $f : \mathcal{C} \to \mathbf{Y}$, there exists an L-Lipschitz function $F : \mathcal{M} \to \mathbf{Y}$ whose restriction to $\mathcal{C}$ coincides with f. By [47] we have $\mathsf{e}(\mathcal{M}) \lesssim \mathsf{SEP}(\mathcal{M})$, and the forthcoming work [15] demonstrates that $\mathsf{e}(\mathcal{M}) \gtrsim \sqrt{\mathsf{SEP}(\mathcal{M})}$, so $\mathsf{e}(\mathcal{M}) < \infty$ if and only if $\mathsf{SEP}(\mathcal{M}) < \infty$. A substitution of the former upper bound on $\mathsf{e}(\mathcal{M})$ provides the following extension theorem:

▶ **Theorem 2.** *If $\mathbf{X} = (\mathsf{M}_n(\mathbb{R}), \|\cdot\|_{\mathbf{X}})$ is a unitarily invariant normed space, then:*

$$\mathsf{e}(\mathbf{X}) \lesssim \sqrt{n}\|\mathsf{I}_n\|_{\mathbf{X}}\operatorname{diam}(B_{\mathbf{X}}).$$

The special case $\mathbf{X} = \mathsf{S}_\infty^n$ of Theorem 2 asserts that for every subset $\mathcal{C}$ of $\mathsf{M}_n(\mathbb{R})$ and every Banach space $(\mathbf{Y}, \|\cdot\|_{\mathbf{Y}})$, if $f : \mathcal{C} \to \mathbf{Y}$ is 1-Lipschitz with respect to the operator norm (and the given norm $\|\cdot\|_{\mathbf{Y}}$ on the target $\mathbf{Y}$), then it can be extended to a $\mathbf{Y}$-valued function that is defined on all of $\mathsf{M}_n(\mathbb{R})$ and is L-Lipschitz with respect to the operator norm, where $L \lesssim n$. The previously best-known [55] upper bound on L here was $L \lesssim n\sqrt{\log n}$, and the corresponding best impossibility result that is currently available [55] is that there exist suitably chosen $\mathcal{C}, \mathbf{Y}, f$ as above for which necessarily $L \gtrsim \sqrt{n}$. Thus, we now know that:

$$\sqrt[4]{\dim(\mathsf{S}_\infty^n)} = \sqrt{n} \lesssim \mathsf{e}(\mathsf{S}_\infty^n) \lesssim n = \sqrt{\dim(\mathsf{S}_\infty^n)}. \tag{7}$$

▶ **Problem 3.** It remains an interesting open question to determine the growth rate of $\mathsf{e}(\mathsf{S}_\infty^n)$ as $n \to \infty$. For that matter, there is no unitarily invariant normed space $\mathbf{X} = (\mathsf{M}_n(\mathbb{R}), \|\cdot\|_{\mathbf{X}})$ for which the value of $\mathsf{e}(\mathbf{X})$ is known up to $O(1)$ factors, or even up to lower-order factors. Since $\operatorname{diam}(B_{\mathbf{X}}) \geqslant 2\|\mathsf{I}_n\|_{\mathsf{S}_2^n}/\|\mathsf{I}_n\|_{\mathbf{X}} = 2\sqrt{n}\|\mathsf{I}_n\|_{\mathbf{X}}$, Theorem 2 does not provide an upper bound on $\mathsf{e}_n(\mathbf{X})$ that is $o(n)$. At the same time, the general lower bound on $\mathsf{e}(\mathbf{X})$ in [55, Theorem 96] is at most $\sqrt{n}$, since the Banach–Mazur distance between $\mathbf{X}$ and an n^2-dimesnional Hilbert space is at most $\sqrt{n}$, as seen by applying John's theorem [33] to the symmetric space $\mathbf{E} = (\mathbb{R}^n, \|\cdot\|_{\mathbf{E}})$ for which $\mathbf{X} = \mathsf{S}_{\mathbf{E}}^n$. Thus, the bounds in (7) are the best one could hope for using the current state of the art even when S_∞^n is replaced by any unitarily invariant norm on $\mathsf{M}_n(\mathbb{R})$. It seems that a substantial new idea will be needed in order to improve either the upper or lower estimate in (7).

As another application of Theorem 1, if we are given oracle access to (approximate) norm computations in a unitarily invariant normed space $\mathbf{X} = (\mathsf{M}_n(\mathbb{R}), \|\cdot\|_{\mathbf{X}})$, then we obtain a (deterministic) algorithm that evaluates its separation modulus $\mathsf{SEP}(\mathbf{X})$ up to universal constant factors in oracle polynomial time.[3]

Theorem 1 reduces the above task to devising an algorithm that outputs a $O(1)$-factor approximation to $\mathrm{diam}(B_{\mathbf{X}})$ in oracle polynomial time. Write $\mathbf{X} = \mathsf{S}_{\mathbf{E}}^n$ for a symmetric normed space $\mathbf{E} = (\mathbb{R}^n, \|\cdot\|_{\mathbf{E}})$. The S_2^n-diameter of $B_{\mathbf{X}}$ is equal to the diameter $\mathrm{diam}_{\ell_2^n}(B_{\mathbf{E}})$ of $B_{\mathbf{E}} \subseteq \mathbb{R}^n$ in ℓ_2^n. And, $\mathbf{X}$ has the desired oracle if and only if $\mathbf{E}$ does.[4] So, the goal is obtaining an oracle polynomial time $O(1)$-factor approximation algorithm to $\mathrm{diam}_{\ell_2^n}(B_{\mathbf{E}})$.

A reduction to efficient Euclidean diameter estimation should give one pause, as the latter task cannot be achieved for general convex bodies, i.e., there is no oracle polynomial time algorithm that outputs a $O(1)$-factor approximation to the ℓ_2^n-diameter of an arbitrary convex body K in $\mathbb{R}^n$ that is given by a weak membership oracle; for deterministic algorithms, this impossibility result was proved in [9], and randomized algorithms were ruled out in [17, 18]. Such hardness of approximation is also proved in [18] for the problem of computing the diameter $\mathrm{diam}_{\ell_p^n}(\cdot)$ in ℓ_p^n for every $1 \leqslant p < \infty$ (see Remark 10 below for a description of the known inapproximability factors). Nevertheless, in the presence of additional symmetries, we will prove the following theorem, which yields the aforementioned algorithm for approximating $\mathsf{SEP}(\mathbf{X})$ using Theorem 1:

▶ **Theorem 4.** *For every $n \in \mathbb{N}$, $D \geqslant 1$ and $0 < \delta \leqslant \frac{1}{2}$, there is $d = d(D, \delta) \in \mathbb{N}$ satisfying*

$$d \lesssim D + \frac{\log\left(\frac{1}{\delta}\right)}{\delta}, \tag{8}$$

such that for every $1 \leqslant p \leqslant \infty$ there exists a deterministic algorithm Alg_p that takes as input n, D, ε and a symmetric normed space $\mathbf{E} = (\mathbb{R}^n, \|\cdot\|_{\mathbf{E}})$ whose unit ball is given by a weak membership oracle and satisfies

$$\frac{1}{e^{n^D}} \leqslant \|e_1\|_{\mathbf{E}} = \ldots = \|e_n\|_{\mathbf{E}} \leqslant e^{n^D}. \tag{9}$$

The algorithm Alg_p outputs in oracle time n^d a number $\mathrm{Alg}_p(n, D, \delta, \mathbf{E})$ that satisfies

$$(1-\delta)\,\mathrm{diam}_{\ell_p^n}(B_{\mathbf{E}}) \leqslant \mathrm{Alg}_p(n, D, \delta, \mathbf{E}) \leqslant (1+\delta)\,\mathrm{diam}_{\ell_p^n}(B_{\mathbf{E}}). \tag{10}$$

1.1 Reverse Faber–Krahn

The new ingredient of our proof of Theorem 2 is an asymptotic evaluation of the spectral gap of the Laplacian with Dirichlet boundary conditions on the unit ball $B_{\mathbf{X}} \subseteq \mathsf{M}_n(\mathbb{R})$ of any unitarily invariant norm $\mathbf{X} = (\mathsf{M}_n(\mathbb{R}), \|\cdot\|_{\mathbf{X}})$. We will next describe this; all of the relevant (classical and rudimentary) background on the spectral properties of the Dirichlet Laplacian can be found in [60, 21].

[3] A weak membership oracle suffices here (one can obtain from such a membership oracle a norm evaluation oracle by binary search of possible upper bounds on the norm of the queried vector). See [30] for the relevant (standard) background in algorithmic convex geometry. In particular, "oracle polynomial time" means that the algorithm consists of a binary Turing machine that is augmented by the given oracle, and its running time is polynomial in the usual sense, with the understanding that the time required by each call to the oracle is only what is needed to write the question onto, and read the answer from, a tape of the Turing machine. Thus, here we demand that the algorithm outputs a number that is guaranteed to be bounded from above and from below by positive universal constant multiples of $\mathsf{SEP}(\mathbf{X})$, yet it performs only $n^{O(1)}$ norm evaluations.

[4] We will use here only the trivial direction of this equivalence, which is simply restricting the oracle for $\mathbf{X}$ to diagonal matrices. The reverse direction incurs the $n^{O(1)}$- time overhead of computing the singular values of the query matrix.

Fix $n \in \{2,3,\ldots\}$. For a bounded domain $\Omega \subseteq \mathbb{R}^n$, let $\lambda(\Omega)$ be the smallest $\lambda > 0$ such that there exists a function $f : \Omega \to \mathbb{R}$ that is smooth on the interior of Ω, vanishes on the boundary $\partial\Omega$ of Ω, and satisfies $\Delta f = -\lambda f$ on the interior of Ω, where $\Delta = \sum_{i=1}^n \partial^2/(\partial x_i^2)$ is the standard Laplacian on $\mathbb{R}^n$. Set $\lambda(\mathbf{Y}) = \lambda(B_{\mathbf{Y}})$ for each normed space $\mathbf{Y} = (\mathbb{R}^n, \|\cdot\|_{\mathbf{Y}})$. Here, we obtain the following result:

▶ **Theorem 5.** *Every unitarily invariant normed space* $\mathbf{X} = (\mathsf{M}_n(\mathbb{R}), \|\cdot\|_{\mathbf{X}})$ *satisfies:*

$$\lambda(\mathbf{X}) \asymp n^3 \|\mathsf{I}_n\|_{\mathbf{X}}^2. \tag{11}$$

In the setting of Theorem 5, the following asymptotic equivalence is a different way to express (11):

$$\lambda(B_{\mathbf{X}}) \mathrm{vol}_{\dim(\mathbf{X})}^{\frac{2}{\dim(\mathbf{X})}}(B_{\mathbf{X}}) \asymp \dim(\mathbf{X}), \tag{12}$$

where for $m \in \mathbb{N}$ and $0 \leqslant \alpha \leqslant m$ the α-dimensional Hausdorff measure on $\mathbb{R}^m$ is denoted $\mathrm{vol}_\alpha(\cdot)$; the only values of α that occur herein are $\alpha \in \{m, m-1\}$ (Lebesgue measure and surface area measure, respectively). To see that (12) and (11) coincide, note that $\dim(\mathbf{X}) = n^2$ and by [65, equation (2.2)] we have:

$$\mathrm{vol}_{n^2}(B_{\mathbf{X}})^{\frac{1}{n^2}} \asymp \frac{1}{\sqrt{n}\|\mathsf{I}_n\|_{\mathbf{X}}}. \tag{13}$$

The Faber–Krahn inequality [23, 42] (originally conjectured in [61]) states that the left hand side of (12) is minimized when $B_{\mathbf{X}}$ is replaced by the unit ball of the standard Euclidean metric (in our case S_2^n, though the analogous statement holds if $\mathsf{M}_n(\mathbb{R})$ is replaced by $\mathbb{R}^m$ of any dimension). By a direct computation (see e.g. [55, page 32]), this classical extremal statement implies that the left hand side of (12) is bounded from below by a positive universal constant multiple of $\dim(\mathbf{X})$.

The new content of (12) is thus the reverse inequality, i.e., the estimate $\lambda(\mathbf{X}) \lesssim n^3\|\mathsf{I}_n\|_{\mathbf{X}}^2$ in Theorem 5. This confirms for (unit balls of) unitarily invariant matrix norms the following much more general (and still open) reverse Faber–Krahn phenomenon that was conjectured in [55]:

▶ **Conjecture 6** (reverse Faber–Krahn)**.** *For every origin-symmertric convex body* $K \subseteq \mathbb{R}^m$ *(thus,* $x \in K$ *if and only if* $-x \in K$*) there exists a volume-preserving linear transformation* $S \in \mathsf{SL}_m(\mathbb{R})$ *with* $\lambda(SK)\mathrm{vol}_m(K)^{2/m} \asymp m$*.*

Theorem 5 demonstrates that Conjecture 6 holds for any unitarily invariant norm on $\mathsf{M}_n(\mathbb{R}) \cong \mathbb{R}^{n^2}$ with the matrix S being the (n^2-by-n^2) identity on $\mathsf{M}_n(\mathbb{R})$; this too was conjectured in [55] (see Conjecture 50 there, as well as [55, Problem 44] and [55, Remark 172]).

1.2 Weak isomorphic reverse isoperimetry

It was proved in [55] that Theorem 5 implies Theorem 1. The link between these two results is Theorem 7 below. For its formulation, recall that the isoperimetric quotient of a convex body $K \subseteq \mathbb{R}^m$ is defined to be $\mathrm{iq}(K) = \mathrm{vol}_{m-1}(\partial K)/\mathrm{vol}_m(K)^{1-1/m}$.

▶ **Theorem 7.** *For every* $n \in \mathbb{N}$*, if* $\mathbf{X} = (\mathsf{M}_n(\mathbb{R}), \|\cdot\|_{\mathbf{X}})$ *is a unitarily invariant normed space, then there exists an origin-symmetric convex body* $L \subseteq B_{\mathbf{X}}$ *such that:*

$$\mathrm{vol}_{n^2}(L)^{\frac{1}{n^2}} \gtrsim \mathrm{vol}_{n^2}(B_{\mathbf{X}})^{\frac{1}{n^2}} \qquad \text{yet} \qquad \mathrm{iq}(L) \lesssim n. \tag{14}$$

It was proved in [55, Section 1.6.2] that for each unitarily invariant normed space $\mathbf{X} = (\mathsf{M}_n(\mathbb{R}), \|\cdot\|_{\mathbf{X}})$, the validity of Theorem 5 for $\mathbf{X}$ is equivalent to the validity of Theorem 7 for $\mathbf{X}$; see specifically the explanation why [55, Conjecture 49] and [55, Conjecture 50] are equivalent that appears in [55, page 43], for which equation (1.62) of [55] is crucial input, as well as the convexity and uniqueness of the so-called Cheeger body (see page 31 of [55]), per [2]. Therefore, by proving Theorem 5 we will also establish Theorem 7, but we will next briefly explain the meaning of Theorem 7 and its utility for proving Theorem 1.

For $m \in \mathbb{N}$, the isoperimetric theorem implies that the isoperimetric quotient of a convex body $K \subseteq \mathbb{R}^m$ is at least the isoperimetric quotient of the ℓ_2^m-ball. Direct computation shows that the latter is bounded from above and from below by positive universal constant multiples of $\sqrt{m}$. Thus, Theorem 7 says that one can inscribe in $B_{\mathbf{X}}$ an origin-symemtric convex body L that is large in the sense that the first inequality in (14) holds, yet its isoperimetric quotient is as small as possible up to universal constant factors, i.e., $\mathrm{iq}(L)$ has the same order of magnitude as the isoperimetric quotient of the Hilbert–Schmidt unit ball in $\mathsf{M}_n(\mathbb{R})$. This confirms for (unit balls of) unitarily invariant matrix norms the following much more general (and still open) phenomenon that was conjectured in [55]:

▶ **Conjecture 8** (weak isomorphic reverse isoperimetry).[5] *For every origin-symmetric convex body $K \subseteq \mathbb{R}^m$ there exist $\mathsf{S} \in \mathsf{SL}_m(\mathbb{R})$ and an origin-symmetric convex body $L \subseteq \mathsf{S}K$ that satisfies:*

$$\mathrm{vol}_m(L)^{\frac{1}{m}} \gtrsim \mathrm{vol}_m(B_{\mathbf{X}})^{\frac{1}{m}} \qquad \text{yet} \qquad \mathrm{iq}(L) \lesssim \sqrt{m}.$$

Analogously to the discussion in Section 1.1, Theorem 7 demonstrates that Conjecture 8 holds for any unitarily invariant norm on $\mathsf{M}_n(\mathbb{R})$ with the matrix S being the identity on $\mathsf{M}_n(\mathbb{R})$; this too was conjectured in [55] (see specifically Conjecture 49 there).

By [55], the upper bound on $\mathsf{SEP}(\mathbf{X})$ of Theorem 1 (which, as we explained earlier, is the part of Theorem 1 that remained to be proved because the matching lower bound is due to [55]) follows from Theorem 7; specifically, combine [55, Corollary 79] with equation (2.2) of [65], which is reproduced as equation (6.131) in [55]. The resulting construction in [55] of the random partition of $\mathbf{X}$ can be illustrated informally as follows. Let $L \subseteq B_{\mathbf{X}}$ be the auxiliary convex body that Theorem 1 provides. Iteratively remove random translates[6] of L from $\mathsf{M}_n(\mathbb{R})$; this yields almost surely a random partition of $\mathsf{M}_n(\mathbb{R})$ into sets (translates of L minus the union of those translates from the preceding steps of this random iterative removal) that are of $\mathbf{X}$-diameter at most 1, as $L \subseteq B_{\mathbf{X}}$, and [55] demonstrates that the desired estimate on the separation probability holds thanks to the upper bound on the isoperimetric quotient of L in (14).

▶ Remark 9. While the above procedure yields a random partition of $\mathbf{X}$ whose separation modulus is optimal up to universal constant factors, it is not entirely constructive. From the algorithmic perspective it suffices to bound $\mathsf{SEP}(\mathbf{X})$ from above, per Theorem 4 and the discussion that precedes its statement, as by [45] there is a randomized polynomial time algorithm that takes as input any finite subset $\mathcal{C}$ of $\mathbf{X}$ and outputs a $2\mathsf{SEP}(\mathbf{X})$-separating partition of $\mathcal{C}$. One could hope to extract more algorithmic information by assuming that

[5] The adjective "weak" was used in [55] to describe this conjecture because in [55] a stronger phenomenon was also conjectured; we do not need to recall that stronger conjecture because the present work does not address it, and furthermore its weaker variant that we establish herein suffices for proving Theorem 1.

[6] One must define a suitable distribution of those random shifts here; for the purpose of the present sketch, one could aim instead to obtain a separated partition of an arbitrarily large bounded domain in $\mathsf{M}_n(\mathbb{R})$, in which case the shifts can be chosen Lebesgue-uniformly at random.

$B_{\mathbf{X}}$ has a weak membership oracle and aiming to obtain a convex body L as in Theorem 1 which also has a weak membership oracle that is allowed to make $n^{O(1)}$ calls to the oracle of $B_{\mathbf{X}}$ and $n^{O(1)}$ additional Turing machine operations. We do not currently have this because the body L of Theorem 1 is defined implicitly as the solution of a variational problem. We expect that the aforementioned task could be achieved by computing in oracle polynomial time a sufficiently good approximation to the first nontrivial eigenfunction $\varphi : B_{\mathbf{X}} \to \mathbb{R}$ of the Dirichlet Laplacian on $B_{\mathbf{X}}$, i.e., φ is smooth on the interior of $B_{\mathbf{X}}$, it vanishes on $\partial B_{\mathbf{X}}$, and $-\Delta\varphi = \lambda(\mathbf{X})\varphi$ point-wise. By [14] we know that φ is a log-concave function, whence its sub-level sets are convex. We could then repeat the above procedure with L replaced a suitable sub-level set of φ. We conjecture that this will yield an optimal separating partition of (a large bounded domain of) $\mathbf{X}$, and it remains open to implement the above strategy efficiently.

The above proposal is an (especially interesting) instance of a general program to estimate efficiently (in oracle polynomial time) solutions of geometrically meaningful partial differential equations; another important example is the first nontrivial eigenfunction of the Laplacian on a convex body with Neumann boundary conditions, which controls the Kannan–Lovász–Simonovits conjecture [36]; see [52, 48].

2 Proof of Theorem 5

Proof. Fix $n \in \mathbb{N}$. It was proved in [55, Remark 172] that if Theorem 7 holds for S_∞^n, then Theorem 7 holds for every unitarily invariant normed space $\mathbf{X} = (\mathsf{M}_n(\mathbb{R}), \|\cdot\|_{\mathbf{X}})$. By the equivalence of Theorem 5 and Theorem 7 that was proved in [55] and we discussed in Section 1.2, it therefore suffices to prove Theorem 5 when $\mathbf{X} = \mathsf{S}_\infty^n$, where we recall that S_∞^n is $\mathsf{M}_n(\mathbb{R})$ equipped with the $\ell_2^n \to \ell_2^n$ operator norm. So, our goal in the rest of this section will be to demonstrate that $\lambda(\mathsf{S}_\infty^n) \asymp n^3$. Only the upper bound

$$\lambda(\mathsf{S}_\infty^n) \lesssim n^3 \tag{15}$$

remains to be proved, thanks to the Faber–Krahn inequality, as we explained in Section 1.1.

For every $p, n \in \mathbb{N}$ define a function $f_n^p : B_{\mathsf{S}_\infty^n} \to [0,\infty)$ by setting:

$$\forall \mathsf{A} \in B_{\mathsf{S}_\infty^n}, \qquad f_n^p(\mathsf{A}) \stackrel{\mathrm{def}}{=} \det\left(\mathsf{I}_n - \mathsf{A}\mathsf{A}^*\right)^p = \prod_{k=1}^n \left(1 - s_k(\mathsf{A})^2\right)^p. \tag{16}$$

Then, f_n^p is smooth (in fact, it is a polynomial in the entries of A), and the second equality in (16) makes it evident that f_n^p vanishes on $\partial B_{\mathsf{S}_\infty^n} = \{\mathsf{A} \in \mathsf{M}_n(\mathbb{R}) : \ s_1(\mathsf{A}) = 1\}$. The standard (see e.g. [21]) Rayleigh quotient characterization of the smallest nonzero Dirichlet eigenvalue of $(-\Delta)$ therefore gives:

$$\lambda(\mathsf{S}_\infty^n) \leqslant \frac{\int_{B_{\mathsf{S}_\infty^n}} \|\nabla f_n^p(\mathsf{A})\|_{\mathsf{S}_2^n}^2 \,\mathrm{d}\mathsf{A}}{\int_{B_{\mathsf{S}_\infty^n}} f_n^p(\mathsf{A})^2 \,\mathrm{d}\mathsf{A}}. \tag{17}$$

Theorem 5 will thus be proven if we will evaluate exactly the right hand side of (17) as follows:

$$\frac{\int_{B_{\mathsf{S}_\infty^n}} \|\nabla f_n^p(\mathsf{A})\|_{\mathsf{S}_2^n}^2 \,\mathrm{d}\mathsf{A}}{\int_{B_{\mathsf{S}_\infty^n}} f_n^p(\mathsf{A})^2 \,\mathrm{d}\mathsf{A}} = \frac{pn^2(4p+n)^2}{2(2p-1)(4p+1)}. \tag{18}$$

Indeed, the right hand side of (18) is bounded from above and from below by positive universal constant multiples of $n^2 \max\{n^2/p, p\}$. Therefore, up to universal constant factors setting $p = n$ minimizes the right hand side of (18). For this choice of p, the Rayleigh quotient estimate (17) becomes $\lambda(\mathsf{S}^n_\infty) \lesssim n^3$, which is our current goal (15). Observe that the above reasoning shows that any choice of p other than $p \asymp n$ does not yield the desired estimate (15) . The rest of this section will be devoted to establishing (18).

Given $n \in \mathbb{N}$, we will denote by $V_n : \mathbb{R}^n \to [0,\infty)$ the following (Vendermunde determinant) function:

$$\forall x = (x_1,\ldots,x_n) \in \mathbb{R}^n, \qquad V_n(x) \stackrel{\mathrm{def}}{=} \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} |x_i - x_j|. \tag{19}$$

Define a probability density $\mathfrak{J}^p_n : [0,1]^n \to [0,\infty)$ by setting:

$$\forall x = (x_1,\ldots,x_n) \in [0,1]^n, \qquad \mathfrak{J}^p_n(x) \stackrel{\mathrm{def}}{=} \frac{V_n(x)}{Z_n(p)} \prod_{r=1}^{n} \frac{x_r^{2p}}{\sqrt{1-x_r}}, \tag{20}$$

where $Z_n(p) > 0$ denotes the normalization factor such that $\int_{[0,1]^n} \mathfrak{J}^p_n(x)\,\mathrm{d}x = 1$, i.e.,

$$Z_n(p) \stackrel{\mathrm{def}}{=} \int_{[0,1]^n} V_n(x) \prod_{r=1}^{n} \frac{x_r^{2p}}{\sqrt{1-x_r}}\,\mathrm{d}x = \frac{n!}{\pi^{\frac{n}{2}}} \prod_{s=1}^{n} \frac{\Gamma\big(2m + \frac{1}{2}(s+1)\big)\Gamma\big(\frac{1}{2}s\big)^2}{\Gamma\big(2m + \frac{1}{2}(n+s+1)\big)}. \tag{21}$$

The second equality in (21) is a special case of the classical Selberg integral formula [66]. The parameter $Z_n(p)$ is commonly denoted in the literature (see e.g. [27]) as $S_n(\alpha,\beta,\gamma)$ for the setting of parameters $(\alpha,\beta,\gamma) = (2p+1, 1/2, 1/2)$; using the ad hoc shorter notation $Z_n(p)$ is beneficial here since we will not need to consider probability densities other than $\mathfrak{J}^p_n$. Also, we chose to use the letter $\mathfrak{J}$ for the probability density on $[0,1]^n$ in (20) to indicate that it belongs to the 3-parameter family of densities of Jacobi random n-by-n matrix ensembles; specifically, using the parametrization of [3, equation (4.1.5)], the relevant setting of parameters in the present case are $(\beta, r, s) = (1, 4p+1, 0)$.

There exists $c_n > 0$ such that for every Borel-measurable function $f : [0,\infty)^n \to [0,\infty)$ that is invariant under permutations of the coordinates, i.e., $f(s_{\pi(1)},\ldots,s_{\pi(n)}) = f(s)$ for every permutation $\pi \in S_n$ and every point $s = (s_1,\ldots,s_n) \in [0,\infty)^n$, the following identity holds:

$$\int_{\mathsf{M}_n(\mathbb{R})} f\big(s_1(\mathsf{A}),\ldots,s_n(\mathsf{A})\big)\,\mathrm{d}A = c_n \int_{[0,\infty)^n} f(s) \prod_{i=1}^{n-1} \prod_{j=s+1}^{n} \big|s_i^2 - s_j^2\big|\,\mathrm{d}s. \tag{22}$$

(22) is a special case of the Weyl integration formula [74]; see e.g. [3, Proposition 4.1.3]. The exact value of the parameter c_n will not have any role in the ensuing reasoning (it will cancel out in our computations), but it is worthwhile to note in passing that $c_n^{1/n^2} \asymp 1/\sqrt{n}$; see e.g. [26] for much more on such matters. Thanks to (22), the denominator in the left hand side of (18) can be evaluated as follows:

$$\int_{B_{\mathsf{S}^n_\infty}} f_n^p(\mathsf{A})^2\,\mathrm{d}\mathsf{A} \stackrel{(16)\wedge(22)}{=} c_n \int_{[0,1]^n} \prod_{r=1}^{n} \big(1-s_r^2\big)^{2p} \prod_{i=1}^{n-1} \prod_{j=s+1}^{n} \big|s_i^2 - s_j^2\big|\,\mathrm{d}x \stackrel{(21)}{=} \frac{c_n}{2^n} Z_n(p), \tag{23}$$

where for the last step of (23) we applied the change of variable $s = (\sqrt{1-x_1},\ldots,\sqrt{1-x_n})$, whose Jacobian is $(-1)^n/(2^n \prod_{r=1}^n \sqrt{1-x_r})$.

We need to understand the Hilbert–Schmidt norm of the gradient of $(\mathsf{A} \in B_{\mathsf{S}_\infty^n}) \mapsto f_n^p(\mathsf{A})$. For this, as a special case of [51, Theorem 7.1] we see that if we define $g_n^p : \mathbb{R}^n \to \mathbb{R}$ by

$$\forall x \in \mathbb{R}^n, \qquad g_n^p(x) \stackrel{\text{def}}{=} \prod_{i=1}^n \big(1 - x_i^2\big)^p, \tag{24}$$

then for any matrix A in the interior of $B_{\mathsf{S}_\infty^n}$ there are $\mathsf{U} = \mathsf{U}_{\mathsf{A},n,p}, \mathsf{V} = \mathsf{V}_{\mathsf{A},n,p} \in \mathsf{O}_n$ such that

$$\nabla f_n^p(\mathsf{A}) = \mathsf{U} \begin{pmatrix} \frac{\partial g_n^p}{\partial x_1}\big(s_1(\mathsf{A}), \dots, s_n(\mathsf{A})\big) & 0 & \dots & 0 \\ 0 & \frac{\partial g_n^p}{\partial x_2}\big(s_1(\mathsf{A}), \dots, s_n(\mathsf{A})\big) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{\partial g_n^p}{\partial x_n}\big(s_1(\mathsf{A}), \dots, s_n(\mathsf{A})\big) \end{pmatrix} \mathsf{V}.$$

By substituting the definition (24) of g_n^p, we thus have:

$$\nabla f_n^p(\mathsf{A}) = 2p \prod_{i=1}^n \big(1 - s_i(\mathsf{A})^2\big)^p \mathsf{U} \begin{pmatrix} \frac{s_1(\mathsf{A})}{1-s_1(\mathsf{A})^2} & 0 & \dots & 0 \\ 0 & \frac{s_2(\mathsf{A})}{1-s_2(\mathsf{A})^2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{s_n(\mathsf{A})}{1-s_n(\mathsf{A})^2} \end{pmatrix} \mathsf{V}.$$

Since U and V are orthogonal matrices, we therefore have:

$$\|\nabla f_n^p(\mathsf{A})\|_{\mathsf{S}_2^n}^2 = 4p^2 \prod_{r=1}^n \big(1 - s_r(\mathsf{A})^2\big)^{2p} \sum_{k=1}^n \frac{s_k(\mathsf{A})^2}{\big(1 - s_k(\mathsf{A})^2\big)^2}. \tag{25}$$

Consequently, the numerator in the left hand side of (18) can be evaluated as follows:

$$\begin{aligned} &\int_{B_{\mathsf{S}_\infty^n}} \|\nabla f_n^p(\mathsf{A})\|_{\mathsf{S}_2^n}^2 \,\mathrm{d}\mathsf{A} \\ &\quad \stackrel{(22)\wedge(25)}{=} 4p^2 c_n \int_{[0,1]^n} \left(\sum_{k=1}^n \frac{s_k^2}{\big(1 - s_k^2\big)^2} \right) \prod_{r=1}^n \big(1 - s_r^2\big)^{2p} \prod_{i=1}^{n-1} \prod_{j=s+1}^n \big|s_i^2 - s_j^2\big| \,\mathrm{d}x \\ &\quad \stackrel{(20)}{=} \frac{c_n Z_n(p)}{2^n} 4p^2 \int_{[0,1]^n} \sum_{k=1}^n \frac{1 - x_k}{x_k^2} \mathfrak{J}_n^p(x) \,\mathrm{d}x \\ &\quad = \frac{c_n Z_n(p)}{2^n} 4p^2 n \bigg(\int_{[0,1]^n} \frac{\mathfrak{J}_n^p(x)}{x_n^2} \,\mathrm{d}x - \int_{[0,1]^n} \frac{\mathfrak{J}_n^p(x)}{x_n} \,\mathrm{d}x \bigg), \end{aligned} \tag{26}$$

where the second step of (26) makes the same change of variable $s = (\sqrt{1 - x_1}, \dots, \sqrt{1 - x_n})$ as above. By combining (23) and (26) we obtain the following expression for the Rayleigh quotient in (18):

$$\begin{aligned} \frac{\int_{B_{\mathsf{S}_\infty^n}} \|\nabla f_n^p(\mathsf{A})\|_{\mathsf{S}_2^n}^2 \,\mathrm{d}\mathsf{A}}{\int_{B_{\mathsf{S}_\infty^n}} |f_n^p(\mathsf{A})|^2 \,\mathrm{d}\mathsf{A}} &= 4p^2 \int_{[0,1]^n} \Big(\sum_{s=1}^n \frac{1 - x_s}{x_s^2} \Big) \mathfrak{J}_n^p(x) \,\mathrm{d}x \\ &= 4p^2 n \bigg(\int_{[0,1]^n} \frac{\mathfrak{J}_n^p(x)}{x_n^2} \,\mathrm{d}x - \int_{[0,1]^n} \frac{\mathfrak{J}_n^p(x)}{x_n} \,\mathrm{d}x \bigg). \end{aligned} \tag{27}$$

To evaluate the right hand side of (27), consider the following symmetric polynomial $P \in \mathbb{Q}[x_1, \ldots x_n]$:

$$P(x) \stackrel{\text{def}}{=} \sum_{s=1}^{n} \prod_{t \in [n] \setminus \{s\}} x_t^2 + \frac{2}{3} \sum_{s=1}^{n-1} \sum_{t=s+1}^{n} x_s x_t \prod_{k \in [n] \setminus \{s,t\}} x_k^2, \tag{28}$$

where in (28), and throughout, we use the common notation $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$. Thus,

$$\begin{aligned} &\frac{1}{Z_n(p)} \int_{[0,1]^n} P(x) V_n(x) \prod_{s=1}^{n} \frac{x_s^{2p-2}}{\sqrt{1-x_s}} \, \mathrm{d}x \\ &\qquad \stackrel{(20)}{=} n \int_{[0,1]^n} \frac{\mathfrak{J}_n^p(x)}{x_n^2} \, \mathrm{d}x + \frac{n(n-1)}{3} \int_{[0,1]^n} \frac{\mathfrak{J}_n^p(x)}{x_{n-1} x_n} \, \mathrm{d}x. \end{aligned} \tag{29}$$

The final integral that appears in (29) can be evaluated as follows:

$$\begin{aligned} &\int_{[0,1]^n} \frac{\mathfrak{J}_n^p(x)}{x_{n-1} x_n} \, \mathrm{d}x \\ &\qquad \stackrel{(20)}{=} \frac{1}{Z_n(p)} \int_{[0,1]^n} V_n(x) \Bigg(\prod_{t=1}^{n-2} x_t \Bigg) \prod_{s=1}^{n} \frac{x_s^{2p-1}}{\sqrt{1-x_s}} \, \mathrm{d}x = \frac{(4p+n)(4p+n+1)}{4p(4p+1)}, \end{aligned} \tag{30}$$

where the second equality that appears in (30) is an instantiation of the Aomoto integral formula [6] (see also e.g. [5, Theorem 8.1.2]). To evaluate the integral that appears in the left hand side of (29), observe that the polynomial P that is defined in (28) coincides with the Jack polynomial that is denoted in [27] by $P_\lambda^{(1/\gamma)}$, where $\gamma = 1/2$ and λ is the vector in $\mathbb{Z}^n$ whose first $n-1$ coordinates equal 2 and the whose last coordinate equals 0; this is seen by substituting the aforementioned parameters γ, λ into equations (3.2) and (3.10) of [27]. The Kadell integral formula [35], which appears in [27] as equation (3.13) thus gives:

$$\frac{1}{Z_n(p)} \int_{[0,1]^n} P(x) V_n(x) \prod_{s=1}^{n} \frac{x_s^{2p-2}}{\sqrt{1-x_s}} \, \mathrm{d}x = \frac{n(n+2)(4p+n)(4p+n-2)}{24p(2p-1)}. \tag{31}$$

The first integral that appears in the right hand side of (27) is therefore evaluated as follows:

$$\begin{aligned} &\int_{[0,1]^n} \frac{\mathfrak{J}_n^p(x)}{x_n^2} \, \mathrm{d}x \\ &\qquad \stackrel{(29) \wedge (30) \wedge (31)}{=} \frac{(n+2)(4p+n)(4p+n-2)}{24p(2p-1)} - \frac{(n-1)(4p+n)(4p+n+1)}{12p(4p+1)}. \end{aligned} \tag{32}$$

Now, the second integral that appears in the right hand side of (27) can be evaluated as follows:

$$\int_{[0,1]^n} \frac{\mathfrak{J}_n^p(x)}{x_n} \, \mathrm{d}x \stackrel{(20)}{=} \frac{1}{Z_n(p)} \int_{[0,1]^n} V_n(x) \Bigg(\prod_{t=1}^{n-1} x_t \Bigg) \prod_{s=1}^{n} \frac{x_s^{2p-1}}{\sqrt{1-x_s}} \, \mathrm{d}x = \frac{4p+n}{4p}, \tag{33}$$

where the second step of (33) is another instantiation of the Aomoto integral formula. The desired identity (18) follows by substituting (32) and (33) into (27), and then simplifying the resulting expression. ◀

3 Proof of Theorem 4

Proof. Observe that every $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ satisfies the following estimate:

$$\|x\|_{\mathbf{E}} \overset{(5)}{=} \max_{s\in[n]} \frac{\|x\|_{\mathbf{E}} + \|(-x_1, \ldots, -x_{s-1}, x_s, -x_{s+1}, \ldots, -x_n)\|_{\mathbf{E}}}{2}$$
$$\geqslant \max_{s\in[n]} |x_s| \cdot \|e_s\|_{\mathbf{E}} \overset{(9)}{\geqslant} \frac{\|x\|_{\ell_2^n}}{\sqrt{n}e^{n^D}}, \tag{34}$$

where the penultimate step of (34) is an application of the covexity of $\|\cdot\|_{\mathbf{E}}$. Furthermore:

$$\|x\|_{\mathbf{E}} \leqslant \sum_{s=1}^n |x_s| \cdot \|e_s\|_{\mathbf{E}} \overset{(9)}{\leqslant} e^{n^D} \sum_{s=1}^n |x_s| \leqslant \sqrt{n}e^{n^D} \|x\|_{\ell_2^n}, \tag{35}$$

where the first step of (35) is an application of the triangle inequality for $\|\cdot\|_{\mathbf{E}}$ and last step of (35) is an application of Cauchy–Schwarz. Therefore:

$$\frac{1}{\sqrt{n}e^{n^D}} B_{\ell_2^n} \overset{(35)}{\subseteq} B_{\mathbf{E}} \overset{(34)}{\subseteq} \sqrt{n}e^{n^D} B_{\ell_2^n}. \tag{36}$$

Using the terminology of [30], the inclusions (36) mean that $B_{\mathbf{E}}$ is well–bounded and centered. Since in Theorem 4 we are assuming that $B_{\mathbf{E}}$ is given by a weak membership oracle, the Judin–Nemirovskiĭ theorem [34] (see [30, Section 4.3]), which is a striking application of the Shallow-Cut Ellipsoid Method [34] (see [30, Section 3.3]), implies it is possible to optimize linear functionals efficiently on $B_{\mathbf{E}}$. In particular, since the dual norm $\|\cdot\|_{\mathbf{E}^*}$ is given by $\|x\|_{\mathbf{E}^*} = \max_{y\in B_{\mathbf{E}}} \langle x, y\rangle$ for every $x \in \mathbb{R}^n$, where $\langle\cdot,\cdot\rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ denotes the standard scalar product on $\mathbb{R}^n$, it follows that there is an algorithm $\mathrm{Eval}_{\mathbf{E}^*}$ which takes as input $x \in \mathbb{R}^n$ and outputs in oracle time that is polynomial in both $\log(1/\delta)$ and n^D a number $\mathrm{Eval}_{\mathbf{E}^*}(x) \in [0, \infty)$ that is guaranteed to satisfy the following estimates:

$$\Big(1 - \frac{\delta}{2}\Big)\|x\|_{\mathbf{E}^*} \leqslant \mathrm{Eval}_{\mathbf{E}^*}(x) \leqslant \Big(1 + \frac{\delta}{2}\Big)\|x\|_{\mathbf{E}^*}. \tag{37}$$

Define $q = p/(p-1)$ if $p > 1$, and $q = \infty$ if $p = 1$. By [29, Lemma 2], if we denote

$$K_q^n \overset{\mathrm{def}}{=} \big\{a = (a_1, \ldots, a_n) \in \mathbb{R}^n : \ \|a\|_{\ell_q^n} \leqslant 1 \text{ and } a_1 \geqslant a_2 \geqslant \cdots \geqslant a_n \geqslant 0\big\}, \tag{38}$$

then for every $0 < \delta < 1$ there exists a subset $\mathcal{G}(n, q, \delta)$ of K_q^n such that:

$$\forall x \in K_q^n, \quad d_{\ell_q^n}\big(x, \mathcal{G}(n, q, \delta)\big) = \min_{a\in\mathcal{G}(n,q,\delta)} \|x - a\|_{\ell_q^n} \leqslant \frac{\delta}{2}, \tag{39}$$

i.e., $\mathcal{G}(n, q, \delta)$ is $(\delta/2)$-dense in K_q^n with respect to the ℓ_q^n norm, yet its size satisfies:

$$|\mathcal{G}(n, q, \delta)| \leqslant n^{\frac{2\log\left(\frac{30}{\delta}\right)}{\log\left(1+\frac{\delta}{6}\right)}}. \tag{40}$$

So, by (40) for fixed δ the size of $\mathcal{G}(n, q, \delta)$ grows like a fixed power of n, in marked contrast to the fact (which is a straightforward consequence of volume comparison) that every $(\delta/2)$-dense subset of a unit ball of an n-dimensional normed space must have size at least $(2/\delta)^n$.

The algorithm Alg_p of Theorem 4 evaluates $\mathrm{Eval}_{\mathbf{E}^*}(a)$ for each $a \in \mathcal{G}(n, q, \delta)$ and outputs the number:

$$\mathrm{Alg}_p(n, D, \delta, \mathbf{E}) \overset{\mathrm{def}}{=} 2 \max_{a\in\mathcal{G}(n,q,\delta)} \mathrm{Eval}_{\mathbf{E}^*}(a). \tag{41}$$

The oracle running time of this simple procedure is a polynomial in both $\log(1/\delta)$ and n^D times the size of $\mathcal{G}(n,q,\delta)$, so by (40) it is at most n^d where the degree d satisfies the desired bound (8). Furthermore:

$$\operatorname{diam}_{\ell_p^n}(B_{\mathbf{E}}) = 2\sup_{x\in B_{\mathbf{E}}}\|x\|_{\ell_p^n} = 2\sup_{x\in B_{\mathbf{E}}}\sup_{y\in B_{\ell_q^n}}\langle x,y\rangle = 2\sup_{y\in B_{\ell_q^n}}\sup_{x\in B_{\mathbf{E}}}\langle x,y\rangle = 2\sup_{y\in B_{\ell_q^n}}\|x\|_{\mathbf{E}^*}. \quad (42)$$

Since $\mathcal{G}(n,q,\delta)$ is a subset of $B_{\ell_q^n}$, this implies in particular that the following a priori upper bound on the output of Alg_p holds, which gives the second inequality in the desired conclusion (10) of Theorem 4:

$$\begin{aligned}\mathrm{Alg}_p(n,D,\delta,\mathbf{E}) &\overset{(37)\wedge(41)}{\leqslant} 2\max_{a\in\mathcal{G}(n,q,\delta)}\Big(1+\frac{\delta}{2}\Big)\|a\|_{\mathbf{E}^*}\\ &\leqslant 2\Big(1+\frac{\delta}{2}\Big)\sup_{y\in B_{\ell_q^n}}\|x\|_{\mathbf{E}^*} \overset{(42)}{\leqslant} \Big(1+\frac{\delta}{2}\Big)\operatorname{diam}_{\ell_p^n}(B_{\mathbf{E}}).\end{aligned}$$

Define $\mathcal{D}(n,q,\delta)\subseteq B_{\ell_q^n}$ by:

$$\mathcal{D}(n,q,\delta) \overset{\text{def}}{=} \Big\{\big(\varepsilon_1 a_{\pi(1)},\varepsilon_2 a_{\pi(2)},\ldots,\varepsilon_n a_{\pi(n)}\big) : (\pi,(\varepsilon_1,\ldots,\varepsilon_n),a)\in S_n\times\{-1,1\}^n\times\mathcal{G}(n,q,\delta)\Big\}. \quad (43)$$

Then, $\mathcal{D}(n,q,\delta)$ is $(\delta/2)$-dense in $B_{\ell_q^n}$. Indeed, for each $x=(x_1,\ldots,x_n)\in B_{\ell_q^n}$ fix a permutation $\pi=\pi^x\in S_n$ such that $|x_{\pi(1)}|\geqslant|x_{\pi(2)}|\geqslant\ldots\geqslant|x_{\pi(n)}|$. Denoting $\varepsilon_i=\varepsilon_i^x=\operatorname{sign}(x_i)\in\{-1,1\}$ for each $i\in[n]$, the vector $y=(\varepsilon_{\pi(1)}x_{\pi(1)},\ldots,\varepsilon_{\pi(n)}x_{\pi(n)})=(|x_{\pi(1)}|,\ldots,|x_{\pi(n)}|)$ belongs K_q^n, by (39). Now, use (39) to get a point $a\in\mathcal{G}(n,q,\delta)$ with $\|y-a\|_{\ell_q^n}\leqslant\delta/2$, so that the point $b=(\varepsilon_1 a_{\pi^{-1}(1)},\ldots,\varepsilon_n a_{\pi^{-1}(n)})$ belongs to $\mathcal{D}(n,q,\delta)$ by (43), and satisfies $\|x-b\|_{\ell_q^n}=\|y-a\|_{\ell_q^n}\leqslant\delta/2$.

Observe that the convex hull $\operatorname{conv}(\mathcal{D}(n,q,\delta))$ of $\mathcal{D}(n,q,\delta)$ satisfies the following inclusion:

$$\operatorname{conv}\left(\mathcal{D}(n,q,\delta)\right)\supseteq\Big(1-\frac{\delta}{2}\Big)B_{\ell_q^n}. \quad (44)$$

In fact, more generally, for any normed space $\mathbf{X}=(\mathbb{R}^n,\|\cdot\|_{\mathbf{X}})$, if $\mathcal{C}\subseteq B_{\mathbf{X}}$ is $(\delta/2)$-dense in $B_{\mathbf{X}}$ with respect to the metric induced by $\|\cdot\|_{\mathbf{X}}$, then $\operatorname{conv}(\mathcal{C})\supseteq(1-\delta/2)B_{\mathbf{X}}$. Indeed, suppose contrapositively that there exists $x\in(1-\delta/2)B_{\mathbf{X}}\setminus\operatorname{conv}(\mathcal{C})$. Then, by the separation theorem we can fix $y^*\in\mathbb{R}^n$ with $\|y^*\|_{\mathbf{X}^*}=1$ such that $\langle y^*,x\rangle>\langle y^*,c\rangle$ for every $c\in\mathcal{C}$. By the Hahn–Banach theorem, we can fix $y\in B_{\mathbf{X}}$ such that $\langle y^*,y\rangle=1$. Finally, because $\mathcal{C}$ is $(\delta/2)$-dense in $B_{\mathbf{X}}$, we can fix $c_y\in\mathcal{C}$ such that $\|y-c_y\|_{\mathbf{X}}\leqslant\delta/2$. We now arrive at the desired contradiction as follows:

$$1-\frac{\delta}{2}\geqslant\|x\|_{\mathbf{X}}\geqslant\langle y^*,x\rangle>\langle y^*,c_y\rangle=\langle y^*y\rangle-\langle y^*,c_y-y\rangle\geqslant 1-\|c_y-y\|_{\mathbf{X}}\geqslant 1-\frac{\delta}{2}, \quad (45)$$

where the second and the penultimate steps of (45) use the fact that y^* has norm 1 as a linear functional in $\mathbf{X}^*$, the first step of (45) uses the assumption $x\in(1-\delta/2)B_{\mathbf{X}}$, the third step of (45) uses the assumed property of the separating functional y^* for $c=c_y\in\mathcal{C}$, and the final step of (45) uses the choice of c_y.

We can now justify the first inequality of the desired conclusion (10) of Theorem 4 as follows:

$$\begin{aligned}
\mathrm{Alg}_p(n, D, \delta, \mathbf{E}) &\overset{(37)\wedge(41)}{\geqslant} 2 \max_{a \in \mathcal{G}(n,q,\delta)} \Big(1 - \frac{\delta}{2}\Big) \|a\|_{\mathbf{E}^*} \\
&\overset{(43)}{=} 2 \max_{a \in \mathcal{D}(n,q,\delta)} \Big(1 - \frac{\delta}{2}\Big) \|a\|_{\mathbf{E}^*} \qquad (46) \\
&= 2\Big(1 - \frac{\delta}{2}\Big) \max_{x \in \mathrm{conv}\left(\mathcal{D}(n,q,\delta)\right)} \|x\|_{\mathbf{E}^*} \qquad (47) \\
&\overset{(44)}{\geqslant} 2\Big(1 - \frac{\delta}{2}\Big) \max_{x \in \left(1-\frac{\delta}{2}\right) B_{\ell_q^n}} \|x\|_{\mathbf{E}^*} \\
&= 2\Big(1 - \frac{\delta}{2}\Big)^2 \max_{x \in B_{\ell_q^n}} \|x\|_{\mathbf{E}^*} \\
&\overset{(42)}{=} \Big(1 - \frac{\delta}{2}\Big)^2 \mathrm{diam}_{\ell_p^n}(B_{\mathbf{E}}) \\
&> (1 - \delta)\, \mathrm{diam}_{\ell_p^n}(B_{\mathbf{E}}),
\end{aligned}$$

where (46) is the step in which the assumption that $\|\cdot\|_{\mathbf{E}}$ is a symmetric norm is used, as this implies (by definition) that its dual $\|\cdot\|_{\mathbf{E}^*}$ is also a symmetric norm, and (47) holds since $\|\cdot\|_{\mathbf{E}^*}$ is a convex function. ◀

▶ Remark 10. In the Introduction we stated the contrast between Theorem 4 and the known nonexistence of any oracle polynomial time algorithm that computes a $O(1)$-factor approximation to the ℓ_p^n-diameter of an arbitrary (so, not necessarily symmetric, as in Theorem 4) convex body in $\mathbb{R}^n$ that is given by a weak membership oracle, where $1 \leqslant p < \infty$ is fixed. This contrast is even more severe because the known impossibility results rule out approximation factors that tend to ∞ with n, as we will next recall.

For $n \in \mathbb{N}$, $h \geqslant 3$ and $1 \leqslant p < \infty$, let $D_p^h(n)$ be the smallest $\alpha \geqslant 1$ for which there is a deterministic algorithm that takes as input a convex body $K \subseteq \mathbb{R}^n$ given by a weak membership oracle, and outputs in oracle time at most n^h a number $\Delta_p(K)$ satisfying $\mathrm{diam}_{\ell_p^n}(K) \leqslant \Delta_p(K) \leqslant \alpha\, \mathrm{diam}_{\ell_p^n}(K)$. Similarly, let $R_p^h(n)$ denote the smallest $\alpha \geqslant 1$ for which there is a randomised algorithm that outputs in oracle time at most n^h a number $\Delta_p(K)$ satisfying $\mathrm{diam}_{\ell_p^n}(K) \leqslant \Delta_p(K) \leqslant \alpha\, \mathrm{diam}_{\ell_p^n}(K)$ with probability at least, say, $1/3$.

Using the above notations, for $2 \leqslant p < \infty$ we have $D_p^h(n) \asymp_h R_p^h(n) \asymp_h (n/\log n)^{1/p}$; when $p = 2$ these bounds on $D_2^h(n)$ are due to [9], and these bounds for both $D_p^h(n)$ and $R_p^h(n)$, and arbitrary $2 < p < \infty$, are due to [17, 18]. If $1 \leqslant p < 2$, then $R_p^h(n) \asymp_h \sqrt{n/\log n}$ by [38, 39], improving by unbounded polylog factor over the corresponding bounds of [17, 18]. For deterministic algorithms, when $1 \leqslant p < 2$ we have $D_p^h(n) \lesssim_h \sqrt{n}/(\log n)^{1-1/p}$ by [17, 18], and the best-known lower bound is $D_p^h(n) \gtrsim_h \sqrt{n/\log n}$, which follows from the trivial lower bound $D_p^h(n) \geqslant R_p^h(n)$ and the aforementioned asymptotic evaluation of $R_p^h(n)$. Thus, for fixed $1 \leqslant p < 2$ there remains a gap of $(\log n)^{1/p-1/2}$, which tends to ∞ with n, between the best-known upper and lower bounds on the deterministic approximation ratio $D_p^h(n)$. While it seems reasonable to expect that this gap (noted in [17, 18, 16]) could be closed using available methods, to the best of our knowledge this has not yet been carried out, even though it would be worthwhile to do so.

References

1 Noga Alon, Richard M. Karp, David Peleg, and Douglas B. West. A graph-theoretic game and its application to the k-server problem (extended abstract). In *On-Line Algorithms, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, February 11-13, 1991*, pages 1–10, 1991. doi:10.1090/DIMACS/007/01.

2 François Alter and Vicent Caselles. Uniqueness of the Cheeger set of a convex body. *Nonlinear Anal.*, 70(1):32–44, 2009. doi:10.1016/j.na.2007.11.032.

3 Greg W. Anderson, Alice Guionnet, and Ofer Zeitouni. *An introduction to random matrices*, volume 118 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2010.

4 Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 459–468, 2006. doi:10.1109/FOCS.2006.49.

5 George E. Andrews, Richard Askey, and Ranjan Roy. *Special functions*, volume 71 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1999. doi:10.1017/CBO9781107325937.

6 K. Aomoto. Jacobi polynomials associated with Selberg integrals. *SIAM J. Math. Anal.*, 18(2):545–549, 1987. doi:10.1137/0518042.

7 Baruch Awerbuch and David Peleg. Sparse partitions (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 503–513, 1990. doi:10.1109/FSCS.1990.89571.

8 Nikhil Bansal, Uriel Feige, Robert Krauthgamer, Konstantin Makarychev, Viswanath Nagarajan, Joseph Naor, and Roy Schwartz. Min-max graph partitioning and small set expansion. *SIAM J. Comput.*, 43(2):872–904, 2014. doi:10.1137/120873996.

9 Imre Bárány and Zoltán Füredi. Computing the volume is difficult. *Discrete Comput. Geom.*, 2(4):319–326, 1987. doi:10.1007/BF02187886.

10 Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 184–193. IEEE, 1996. doi:10.1109/SFCS.1996.548477.

11 Yair Bartal. On approximating arbitrary metrices by tree metrics. In *STOC '98 (Dallas, TX)*, pages 161–168. ACM, New York, 1999.

12 Rajendra Bhatia. *Matrix analysis*, volume 169 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1997. doi:10.1007/978-1-4612-0653-8.

13 Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

14 Herm Jan Brascamp and Elliott H. Lieb. On extensions of the Brunn-Minkowski and Prékopa-Leindler theorems, including inequalities for log concave functions, and with an application to the diffusion equation. *J. Functional Analysis*, 22(4):366–389, 1976. doi:10.1016/0022-1236(76)90004-5.

15 Mark Braverman and Assaf Naor. Quantitative Wasserstein rounding. Preprint, 2025.

16 Andreas Brieden. Geometric optimization problems likely not contained in $\mathbb{APX}$. *Discrete Comput. Geom.*, 28(2):201–209, 2002. doi:10.1007/s00454-002-2756-x.

17 Andreas Brieden, Peter Gritzmann, Ravi Kannan, Victor Klee, László Lovász, and Miklós Simonovits. Approximation of diameters: Randomization doesn't help. In *39th Annual Symposium on Foundations of Computer Science, FOCS 1998, Palo Alto, California, USA, November 8-11, 1998*, pages 244–251. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743451.

18 Andreas Brieden, Peter Gritzmann, Ravindran Kannan, Victor Klee, László Lovász, and Miklós Simonovits. Deterministic and randomized polynomial-time approximation of radii. *Mathematika*, 48(1-2):63–105, 2001. doi:10.1112/S0025579300014364.

19 Alexander Brudnyi and Yuri Brudnyi. *Methods of geometric analysis in extension and trace problems. Volume 2*, volume 103 of *Monographs in Mathematics*. Birkhäuser/Springer Basel AG, Basel, 2012.

20 M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 379–388, 1998. doi:10.1109/SFCS.1998.743488.

21 Isaac Chavel. *Eigenvalues in Riemannian geometry*, volume 115 of *Pure and Applied Mathematics*. Academic Press, Inc., Orlando, FL, 1984. Including a chapter by Burton Randol, With an appendix by Jozef Dodziuk.

22 John P. Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, 16(89):2859–2900, 2015. doi:10.5555/2789272.2912091.

23 G. Faber. Beweis, daß unter allen homogenen Membranen von gleicher Fläche und gleicher Spannung die kreisförmige den tiefsten Grundton gibt. Münch. Ber. 1923, 169-172 (1923)., 1923.

24 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):485–497, 2004. doi:10.1016/j.jcss.2004.04.011.

25 Ky Fan. Maximum properties and inequalities for the eigenvalues of completely continuous operators. *Proc. Nat. Acad. Sci. U.S.A.*, 37:760–766, 1951. doi:10.1073/pnas.37.11.760.

26 P. J. Forrester. *Log-gases and random matrices*, volume 34 of *London Mathematical Society Monographs Series*. Princeton University Press, Princeton, NJ, 2010. doi:10.1515/9781400835416.

27 Peter Forrester and Ole Warnaar. The importance of the Selberg integral. *Bulletin of the American Mathematical Society*, 45:489–534, October 2008. doi:10.1090/S0273-0979-08-01221-4.

28 Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 4 edition, 2013.

29 W. T. Gowers. Symmetric block bases in finite-dimensional normed spaces. *Israel J. Math.*, 68(2):193–219, 1989. doi:10.1007/BF02772661.

30 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, second edition, 1993. doi:10.1007/978-3-642-78240-4.

31 Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 534–543, 2003. doi:10.1109/SFCS.2003.1238226.

32 Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 2 edition, 2012.

33 Fritz John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948*, pages 187–204. Interscience Publishers, Inc., New York, N. Y., 1948.

34 D. B. Judin and A. S. Nemirovskiĭ. Informational complexity and effective methods for the solution of convex extremal problems. *Èkonom. i Mat. Metody*, 12(2):357–369, 1976.

35 Kevin W. J. Kadell. The Selberg-Jack symmetric functions. *Adv. Math.*, 130(1):33–102, 1997. doi:10.1006/aima.1997.1642.

36 R. Kannan, L. Lovász, and M. Simonovits. Isoperimetric problems for convex bodies and a localization lemma. *Discrete Comput. Geom.*, 13(3-4):541–559, 1995. doi:10.1007/BF02574061.

37 David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *J. ACM*, 45(2):246–265, 1998. doi:10.1145/274787.274791.

38 Subhash Khot and Assaf Naor. Linear equations modulo 2 and the L_1 diameter of convex bodies [extended abstract]. In *48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007, Providence, RI, USA, October 20-23, 2007, Proceedings*, pages 318–328. IEEE Computer Society, 2007. doi:10.1109/FOCS.2007.20.

39 Subhash Khot and Assaf Naor. Linear equations modulo 2 and the L_1 diameter of convex bodies. *SIAM J. Comput.*, 38(4):1448–1463, 2008. doi:10.1137/070691140.

40 Philip N. Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 682–690, 1993. doi:10.1145/167088.167261.

41 Goran Konjevod, R. Ravi, and F. Sibel Salman. On approximating planar metrics by tree metrics. *Inform. Process. Lett.*, 80(4):213–219, 2001. doi:10.1016/S0020-0190(01)00161-2.

42 E. Krahn. Über Minimaleigenschaften der Kugel in drei und mehr Dimensionen. Acta Univ. Dorpat A 9, 1-44 (1926)., 1926.

43 Robert Krauthgamer and Nir Petruschka. Lipschitz decompositions of finite ℓ_p metrics. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry, SoCG 2025, June 23-27, 2025, Kanazawa, Japan*, volume 332 of *LIPIcs*, pages 66:1–66:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SOCG.2025.66.

44 Robert Krauthgamer, Nir Petruschka, and Shay Sapir. The Power of Recursive Embeddings for ℓ_p Metrics. To appear in 66th IEEE Symposium on Foundations of Computer Science (FOCS 2025). Preprint available at https://arxiv.org/pdf/2503.18508, 2025.

45 Robert Krauthgamer and Tim Roughgarden. Metric clustering via consistent labeling. *Theory Comput.*, 7:49–74, 2011. doi:10.4086/toc.2011.v007a005.

46 J. R. Lee and A. Naor. Metric decomposition, smooth measures, and clustering. Unpublished manuscript, available at https://web.math.princeton.edu/~naor/homepage%20files/cluster.pdf, 2003.

47 James R. Lee and Assaf Naor. Extending Lipschitz functions via random metric partitions. *Invent. Math.*, 160(1):59–95, 2005. doi:10.1007/s00222-004-0400-5.

48 Yin Tat Lee and Santosh S. Vempala. The Kannan-Lovász-Simonovits conjecture. In *Current developments in mathematics 2017*, pages 1–36. Int. Press, Somerville, MA, 2019.

49 Frank Thomson Leighton and Satish Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 422–431, 1988. doi:10.1109/SFCS.1988.21958.

50 Adrian S. Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analysis*, 2(1-2):173–183, 1995.

51 Adrian S. Lewis and Hristo S. Sendov. Nonsmooth analysis of singular values. part i: Theory. *Set-Valued Analysis*, 13:213–241, 2005. URL: https://api.semanticscholar.org/CorpusID:36823342.

52 Emanuel Milman. On the role of convexity in isoperimetry, spectral gap and concentration. *Invent. Math.*, 177(1):1–43, 2009. doi:10.1007/s00222-009-0175-9.

53 L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford Ser. (2)*, 11:50–59, 1960. doi:10.1093/qmath/11.1.50.

54 Assaf Naor. Probabilistic clustering of high dimensional norms. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 690–709. SIAM, Philadelphia, PA, 2017. doi:10.1137/1.9781611974782.44.

55 Assaf Naor. *Extension, separation and isomorphic reverse isoperimetry*, volume 11 of *Memoirs of the European Mathematical Society*. European Mathematical Society (EMS), Berlin, 2024. doi:10.4171/mems/11.

56 Assaf Naor and Kevin Ren. Euclidean embedding, randomized clustering, and Lipschitz extension for finite and doubling subsets of L_p when $p > 2$. Preprint available at https://arxiv.org/pdf/2410.21931, 2025.

57 Assaf Naor and Kevin Ren. Optimal randomized clustering for subsets of L_p when $p > 2$. To appear in 37th ACM-SIAM Symposium on Discrete Algorithms (SODA 2026), 2026.

58 David Peleg and Eilon Reshef. Deterministic polylog approximation for minimum communication spanning trees (extended abstract). In *Automata, languages and programming (Aalborg, 1998)*, volume 1443 of *Lecture Notes in Comput. Sci.*, pages 670–681. Springer, Berlin, 1998. doi:10.1007/BFb0055092.

59 Michal Pešta. Unitarily invariant errors-in-variables estimation. *Statistical Papers*, 57:1041–1057, 2016. doi:10.1007/s00362-016-0800-9.

60 G. Pólya and G. Szegö. *Isoperimetric Inequalities in Mathematical Physics*. Annals of Mathematics Studies, No. 27. Princeton University Press, Princeton, N. J., 1951.

61 John William Strutt Rayleigh, Baron. *The Theory of Sound*. Dover Publications, New York, 1945. 2d ed.

62 Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010. doi:10.1137/070697835.

63 Arvind K. Saibaba. Randomized subspace iteration: Analysis of canonical angles and unitarily invariant norms. *SIAM Journal on Matrix Analysis and Applications*, 40(1):23–48, 2019. doi:10.1137/18M1179432.

64 Robert Schatten. *A Theory of Cross-Spaces*, volume No. 26 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ, 1950.

65 Carsten Schütt. On the volume of unit balls in Banach spaces. *Compositio Math.*, 47(3):393–407, 1982. URL: http://www.numdam.org/item?id=CM_1982__47_3_393_0.

66 Atle Selberg. Remarks on a multiple integral. *Norsk Mat. Tidsskr.*, 26:71–79, 1944.

67 Barry Simon. *Trace Ideals and Their Applications*, volume 120 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2 edition, 2005. doi:10.1090/surv/120.

68 Martin Stoll. A literature survey of matrix methods for data science. *GAMM-Mitteilungen*, 43(3):e202000013, 2020. doi:10.1002/gamm.202000013.

69 David Sutter, Mario Berta, and Marco Tomamichel. Multivariate trace inequalities. *Communications in Mathematical Physics*, 352(1):37–58, 2017. doi:10.1007/s00220-016-2778-5.

70 Marco Tomamichel. *Quantum Information Processing with Finite Resources: Mathematical Foundations*, volume 5 of *SpringerBriefs in Mathematical Physics*. Springer, Cham, 2016. doi:10.1007/978-3-319-21891-5.

71 J. von Neumann. Some matrix inequalities and metrization of matric space. Mitt. Forschungsinst. Math. Mech. Kujbyschew-Univ. Tomsk 1, 286-300 (1937), 1937.

72 John Watrous. *The Theory of Quantum Information*. Cambridge University Press, Cambridge, 2018. doi:10.1017/9781316848142.

73 G. A. Watson. The solution of orthogonal procrustes problems for a family of orthogonally invariant norms. *Advances in Computational Mathematics*, 2:393–405, 1994. doi:10.1007/BF02521606.

74 Hermann Weyl. *The Classical Groups. Their Invariants and Representations*. Princeton University Press, Princeton, NJ, 1939.

75 Kemin Zhou, John C. Doyle, and Keith Glover. *Robust and Optimal Control*. Prentice Hall, Upper Saddle River, NJ, 1996.

Singular Arrange and Traverse Algorithm for Computing Reeb Spaces of Bivariate PL Maps

Petar Hristov[1]
Scientific Visualization Group, Department of Science and Technology (ITN), Linköping University, Sweden

Ingrid Hotz
Scientific Visualization Group, Department of Science and Technology (ITN), Linköping University, Sweden

Talha Bin Masood
Scientific Visualization Group, Department of Science and Technology (ITN), Linköping University, Sweden

Abstract

We present an exact and efficient algorithm for computing the Reeb space of a bivariate PL map. The Reeb space is a topological structure that generalizes the Reeb graph to the setting of multiple scalar-valued functions defined over a shared domain, a situation that frequently arises in practical applications. While the Reeb graph has become a standard tool in computer graphics, shape analysis, and scientific visualization, the Reeb space is still in the early stages of adoption. Although several algorithms for computing the Reeb space have been proposed, none offer an implementation that is both exact and efficient, which has substantially limited its practical use. To address this gap, we introduce *singular arrange and traverse*, a new algorithm built upon the *arrange and traverse* framework [26]. Our method exploits the fact that, in the bivariate case, only singular edges contribute to the structure of Reeb space, allowing us to ignore many regular edges [40]. This observation results in substantial efficiency gains on datasets where most edges are regular, which is common in many numerical simulations of physical systems. We provide an implementation of our method and benchmark it against the original *arrange and traverse* algorithm, showing performance gains of up to four orders of magnitude on real-world datasets.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Geometric topology

Keywords and phrases Computational topology, Reeb graph, Reeb space, Multivariate data, Multi-field, Geometric arrangement

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.57

Related Version *arXiv Version*: `https://arxiv.org/abs/2602.21087`

Supplementary Material *Software (Source Code)*: `https://doi.org/10.5281/zenodo.18759995`

Funding *Ingrid Hotz*: The author acknowledges support from the Swedish e-Science Research Center (SeRC), the ELLIIT environment for strategic research in Sweden, and the Swedish Research Council (VR) grant 2019-05487.
Talha Bin Masood: This work was primarily supported by the WASP-DDLS grant from Wallenberg Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The author acknowledges Swedish Research Council (VR) grant 2023-04806 and the Swedish e-Science Research Center (SeRC) for additional funding support.

Acknowledgements We thank Dr. Nanna H. List (KTH Royal Institute of Technology, Stockholm) for providing the MVK dataset used for evaluation in this paper. The Ethane-diol dataset is taken from the TTK dataset repository (`https://github.com/topology-tool-kit/ttk-data`) with acknowledgments to Roberto Alvarez Boto. The enzo dataset is taken from the *The Enzo Collaboration* with acknowledgments to Michael Norman (University of California, San Diego, USA).

[1] Corresponding author

© Petar Hristov, Ingrid Hotz, and Talha Bin Masood;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

Visualizing and analyzing the structure of scalar-valued functions is fundamental across many disciplines, including physics [38], chemistry [3, 21], atmosphere science [29, 42], biology [32, 34], computer graphics [4, 22], and machine learning [12, 31]. Topological tools such as persistent homology, the Morse-Smale complex, and the Reeb graph are state-of-the-art methods for extracting meaningful features from scalar fields [23]. Their widespread adoption is largely due to the fact that they can be computed *robustly* and *efficiently*.

The multivariate setting, where multiple scalar functions are defined over a shared domain, is considerably more complex. In this context, the interrelated structure between the functions cannot be captured adequately by scalar field techniques. To address this limitation, the Reeb graph can be generalized to the Reeb space [18]. The Reeb space is a quotient space that contracts the components of all preimages of points, also called *fibers*. The Reeb space of a bivariate PL map over a compact PL manifold can be represented as a two-dimensional cell complex [18, 27], where the 2-cells, referred to as sheets, represent features in data.

Four algorithms have been proposed for computing the Reeb space of a PL map. The first algorithm [18] segments the domain into regions where fibers have uniform connectivity and then glues those regions appropriately to form the sheets of the Reeb space. The Jacobi fiber surfaces algorithm [40] significantly improves practical performance in the bivariate case by computing these regions only for the singular (or Jacobi) edges in the domain, where fibers can change their connectivity. The third algorithm uses a hierarchical decomposition-based approach [10] to compute the Reeb space of a bivariate PL map from a sequence of nested Reeb graphs. Finally, the arrange and traverse algorithm [26] computes combinatorial descriptions of fibers called preimage graphs and determines their topological correspondence to obtain regions of uniform fiber connectivity that match the sheets of the Reeb space.

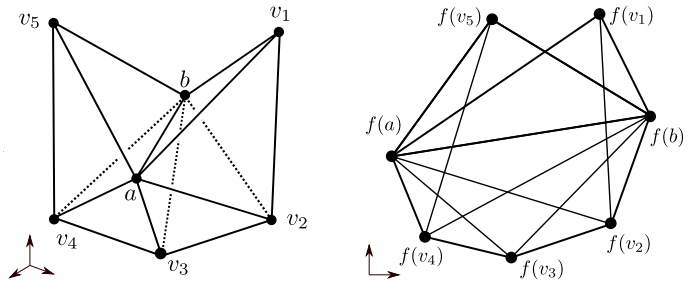
While all four Reeb space algorithms have close to quadratic algorithmic complexity in the bivariate case, their implementations suffer from either efficiency or robustness issues. The arrange and traverse algorithm [26] provides a robust implementation [24] in the bivariate case with CGAL's exact implementation of arrangements of 2D line segments [19]. However, due to its slow running time, it cannot handle real-life datasets without significant downsampling. The Jacobi fiber surfaces algorithm [40] offers better performance, but, as has been noted [10], its implementation is not exact without additional care in handling Jacobi fiber surface intersections. This paper aims to combine the strengths of these two approaches to provide a Reeb space algorithm with an implementation that is both efficient and robust.

We propose the *singular arrange and traverse* algorithm for computing the Reeb space of a generic bivariate PL map $f : |K| \to \mathbb{R}^2$, defined over a triangulation of a 3-manifold K. This is a specialization of the arrange and traverse algorithm that does not require the arrangement of the images of all edges of K in the range, only the singular ones, where the connectivity of the fibers can change. Using red-blue line segment intersection on the images of regular and singular edges, our algorithm computes the Reeb space from a subset of all preimage graphs, which we call *essential*. The worst-case running time of our algorithm is $O(N_s N_t \log N_t)$, where N_s is the number of singular edges and N_t is the number of triangles. This is a significant improvement because datasets with a low proportion of singular edges are common in practice. We provide an exact implementation and benchmark it against the arrange and traverse algorithm, the only other exact implementation known to the authors, achieving up to four orders of magnitude improvement in running time and memory footprint.

This paper is structured as follows: in Section 2, we introduce the background relevant to Reeb spaces of PL maps; in Section 3, we present the singular arrange and traverse algorithm; in Section 4, we prove its correctness and worst-case complexity; in Section 5 we describe our implementation, which we benchmark in Section 6. We conclude in Section 7.

2 Preliminaries

Let K be a triangulation of a 3-manifold and let $f : |K| \to \mathbb{R}^2$ be a piecewise linear (PL) map [35, 17]; see Figure 1. We will refer to f as a *bivariate* PL map because it can be decomposed into two scalar PL maps $f = (f_1, f_2)$, where $f_1, f_2 : |K| \to \mathbb{R}$. We assume that f is *generic*, in the sense that the images of no three vertices are collinear and the images of no three edges intersect at a single point in their interiors [25]. We refer to the images of edges under f in the range as *segments* and to the image of vertices under f as *vertex points*. For an edge e with endpoints a and b, we will also refer to e as ab.



(a) Tetrahedral mesh. **(b)** Image in the range.

Figure 1 Example tetrahedral mesh (a) and its image in the range under a bivariate PL map (b).

The *upper link* of an edge ab is a sub-complex of the link of ab induced by the vertices v in the link, whose vertex points $f(v)$ have a positive orientation with respect to $f(a)$ and $f(b)$ [27]. The *lower link* is defined analogously by the vertex points with negative orientation. In general we will assume that the endpoints of an edge or a segment are ordered lexicographically in order to establish a canonical orientation. For example, the upper link of ab in Figure 1 is $\{v_1, v_5\}$ and the lower link is $\{v_2, v_3, v_4, v_2v_3, v_3v_4\}$.

An edge is *regular* if both its lower and upper links have one simply connected component. Otherwise, that edge is *singular*, or Jacobi [16]. A vertex is *singular* if it is incident to at least one singular edge and *regular* otherwise. We call the images of singular edges and vertices in the range *singular segments and vertex points* respectively. The collection of singular edges and vertices is a one-dimensional subcomplex of K, which we call the *singular set* S_f.

A *fiber* is the preimage of a point $f^{-1}((x_1, x_2)) \subset |K|$ for $(x_1, x_2) \in \mathbb{R}^2$, which we call a *fiber point* (see Figure 2). A simplex intersected by a fiber is called *active* with respect to that fiber. We refer to the connected components of fibers as *fiber components*. Since fiber components can change their connectivity only at the singular set [27, 40], those that intersect S_f are called *singular*, and those that do not are called *regular*. Regular fiber components are polylines, while singular fiber components are either an isolated point or a number of polylines glued together at exactly one or two points [24]. Fibers with only regular components are *regular*, otherwise, they are *singular*.

Singular edges, analogous to critical points in the univariate case, can be categorized in the following way: *definite* edges create or destroy fiber components (analogous to minima and maxima) and *indefinite* edges merge or split fiber components (analogous to saddles) [27, 26]. Indefinite edges that split or merge exactly two components are referred to as *simple*. The singular set is called *simple* [18, 10] when it is a collection of polylines and all indefinite edges are simple. Non-simple singular edges can be unfolded into multiple simple ones (like monkey saddles), but that requires the addition of new simplices to the input mesh [16].

The *Reeb space* R_f of a PL map f is a topological structure that contracts each fiber component to a single point. It is the quotient space of $|K|$ with respect to the equivalence relation $\sim$, where $x \sim y$ if and only if x and y belong to the same fiber component. The

Reeb space is a part of the Stein factorization $f = \pi \circ q$, where $q : |K| \to R_f$ is the quotient map with respect to $\sim$ and π is the natural map such that the diagram commutes [30]. For computational purposes, the Reeb space of a bivariate PL map can be represented as a two-dimensional cell complex whose 2-cells, referred to as *sheets*, are glued together along the image of the singular set under q [9]. For an example see Figure 2.

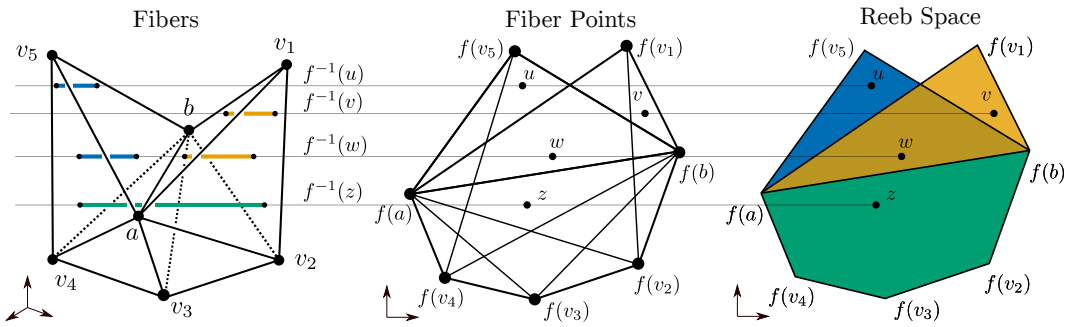


Figure 2 Examples of the change of connectivity of four fibers at the indefinite edge ab and the definite edges av_1 and bv_5. The Reeb space of f has three sheets (blue, yellow and green) – one for each fiber component. To visualize the Reeb space we map its sheets to the range and overlay them.

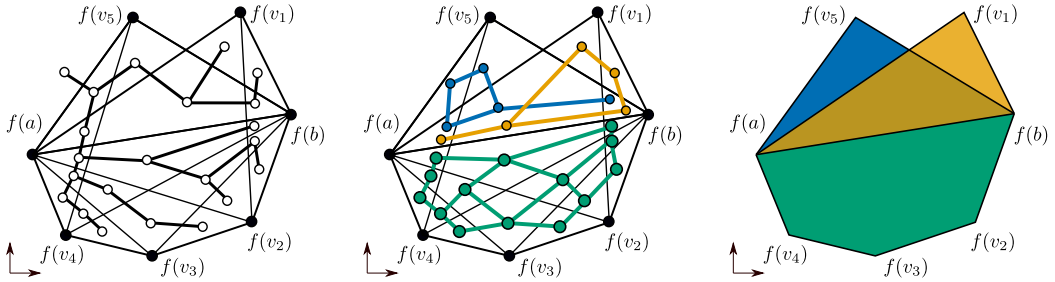
2.1 Arrange and Traverse Algorithm

The Reeb space of a PL map can be computed with four different approaches, which we outlined in Section 1. We will review the *arrange and traverse* algorithm [26] in detail because it forms the foundation for our new approach. The *arrange and traverse* algorithm computes the Reeb space by combinatorially representing fibers and tracking how their connectivity changes as they intersect the edges of K. Fibers are represented combinatorially with a preimage graph, similar to Parsa's optimal Reeb graph algorithm [33]. The *preimage graph* of a fiber $f^{-1}((x_1, x_2)) \subset |K|$ for a point $(x_1, x_2) \in \mathbb{R}^2$, which is not in the image of an edge or vertex, is a graph whose vertices are the active triangles with respect to that fiber and whose edges connect triangles which are both the faces of the same tetrahedron.

In the first stage of this algorithm, the edges of K are mapped to the range via f, and their geometric arrangement A is computed (see Section 2.2). Within each face F of A, all fibers have the same active triangles, so they can all be represented by a single *preimage graph*. In the second stage, the faces of A are traversed in order to compute their preimage graphs. The difference in the connectivity of the preimage graphs of two faces F_1 and F_2, which are incident via a segment $f(ab)$, where ab is an edge of K, is locally determined by the upper and lower link of ab. Assume that we cross from F_1 to F_2 via $f(ab)$ by crossing from the half-plane with negative orientation with respect to $f(a)$ and $f(b)$ to the half-plane with positive orientation. Then all triangles abv, where v is in the lower link of ab, are removed from the preimage graph of F_1, and all triangles abv', where v' is in the upper link of ab, are added to the preimage graph of F_1 to form the preimage graph of F_2. For the opposite direction of travel, all triangles abv are added and all triangles abv' are removed.

Once all preimage graphs are computed, the *correspondence graph* H is constructed. The vertices of the correspondence graph are the fiber components of the preimage graphs of all faces. Edges connect vertices representing fiber components when there is a correspondence between them – that is, when a fiber component passes from one face to another without any change in connectivity. Each sheet in the Reeb space is a collection of faces such that the connected components of the preimage graphs of those faces are in the same connected component in the correspondence graph. The adjacency of the sheets in the Reeb space is determined by the adjacency of the faces that comprise them.

For an example of stages of the arrange and traverse algorithm refer to Figure 3. In the bivariate case, the only geometric computations performed by this algorithm are in 2D – computing the arrangement and orientation tests for upper and lower links. The rest of the computation is entirely combinatorial. This is a significant advantage over other Reeb space algorithms, as it allows the use of only 2D geometric predicates [25]. An exact open-source implementation using CGAL can be found on GitHub [24]. Finally, this algorithm can be extended to higher dimensional domains K^d and ranges $\mathbb{R}^k$ such that $d > k$.



(a) Arrangement traversal. **(b)** Correspondence graph. **(c)** Reeb space.

Figure 3 The key stages of the arrange and traverse algorithm [26] for Figure 1.

2.2 Segment Intersections and Arrangements

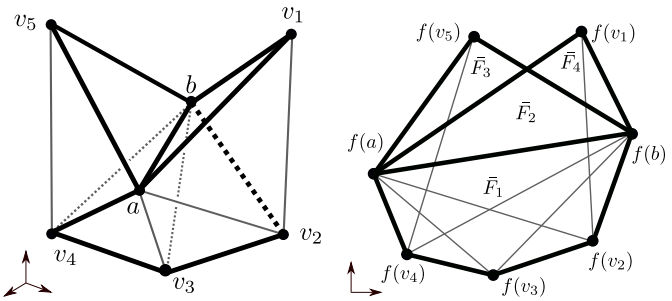
The intersections of n line segments in the plane can be computed in time $O(n \log n + k)$, where k is the number of intersections [11, 2]. A subdivision of the plane by segments into vertices, edges, and faces is called an *arrangement* and can be computed in $O(n \log n + k \log n)$ time [14, 20] and represented with a *doubly connected edge list (DCEL)*. The special case of reporting the intersections between two sets of segments, typically labeled *red* and *blue*, can be solved in $O(n \log n + t)$ time, where t is the number of intersections between the two sets.

3 Singular Arrange and Traverse Algorithm

The *key idea* of the algorithm we propose is that *not all faces and preimage graphs are needed* to compute the Reeb space in the arrange and traverse algorithm (see Section 2.1). The only faces where the connectivity of fibers can change are the ones that are incident to at least one singular segment or singular vertex point. We call these faces *essential* because they outline a part of the boundary of at least one sheet of the Reeb space. More precisely, they outline the images of sheets in the plane under π. We refer to all other faces as *nonessential* because they do not contribute to the shape of the sheets. Our goal is to compute the Reeb space by *using only the essential faces* and their preimage graphs.

3.1 Definitions

Let K be a triangulation of a 3-manifold and let $f : |K| \to \mathbb{R}^2$ be a generic PL map. Let S_f be the singular set of f and let R_f be the Reeb space of f (see Section 2). We *do not* assume that the singular set is simple. For clarity of exposition, we will assume that $f(S_f)$ is connected – we will deal with the case where it is not in Section 3.3. We will refer to preimage graphs as *fiber graphs*, since fibers are defined as preimages of points under f. Let $A = \text{Arrangement}(f(K^{(1)}))$, where $K^{(1)}$ is the 1-skeleton of K, be the arrangement of all segments, and let $\bar{A} = \text{Arrangement}(f(S_f))$ be the arrangement of all singular segments. We will refer to A and $\bar{A}$ as the *full* and *singular* arrangement respectively (see Figure 4).



(a) Singular edges (in bold). **(b)** Singular segments (in bold).

Figure 4 The singular set (a) of Figure 1 and its image in the range (b). The arrangement $\bar{A}$ of the singular segments has four bounded faces, each subdivided by the faces of the full arrangement.

Consider a face of the singular arrangement $\bar{F} \in \bar{A}$. The fibers that correspond to points in the interior of $\bar{F}$ are homeomorphic to one another because they are all regular – no change in connectivity happens to any of them. Since $\bar{A}$ is an arrangement of a subset of the segments used in A, $\bar{F}$ is subdivided into a set of faces in A such that $\bar{F} = \cup_i F_i$, where $F_i \in A$. We describe the connectivity of fibers inside $\bar{F}$ by grouping the fiber graphs of all faces that subdivide $\bar{F}$ into a fiber graph class.

▶ **Definition 1** (Fiber Graph Class). *Let $\bar{F}$ be a face of the singular arrangement $\bar{A}$ and let $\{F_i\}_i$ be the set of faces of the full arrangement that are in $\bar{F}$. The fiber graph class $\bar{G}_{\bar{F}}$ of $\bar{F}$ is the set of fiber graphs $\bar{G}_{\bar{F}} = \{G_{F_i}\}_i$, where G_{F_i} is the fiber graph of F_i.*

Fiber graph classes allow us to *bundle* together fibers with the same topology to examine their connectivity in bulk. For an example see Figure 5. In order to examine that connectivity, we define the notion of a component of a fiber graph class.

▶ **Definition 2** (Fiber Graph Class Component). *Let $\bar{G}_{\bar{F}} = \{G_{F_i}\}_i$ be the fiber graph class of a face $\bar{F}$ of $\bar{A}$. A component $\bar{C}$ of a fiber graph class $\bar{G}_{\bar{F}}$ is a set $\{C_i\}_i$, where each C_i is a connected component of G_{F_i}, such that any two connected components C_i and C_j of $\bar{C}$ correspond to each other (see correspondence in Section 2.1).*

The connected components of fiber graphs within a single fiber graph class component are in the same connected component of the correspondence graph H. In order to determine how the components of fiber graph classes change their topology across the faces of the singular arrangement we define their correspondence.

▶ **Definition 3** (Fiber Graph Class Component Correspondence). *Two components $\bar{C}$ and $\bar{C}'$ of two fiber graph classes $\bar{G}_{\bar{F}}$ and $\bar{G}_{\bar{F}'}$ correspond to each other if there exist two connected components $C \in \bar{C}$ and $C' \in \bar{C}'$ that correspond to each other.*

We can describe the connectivity of fiber graph class components across all faces of the singular arrangement with the singular correspondence graph.

▶ **Definition 4** (Singular Correspondence Graph). *The singular correspondence graph $\bar{H}$ is a graph whose vertices are the set of all fiber graph class components and whose edges connect corresponding ones (see Definition 3).*

The sheets of the Reeb space have a one-to-one matching with the connected components of the singular correspondence graph. The geometry of a sheet of the Reeb space, or its image in the plane via π, is then the union of all faces $\cup_i \bar{F}_i$ whose fiber graph classes have components that span the connected component of $\bar{H}$ that matches that sheet.

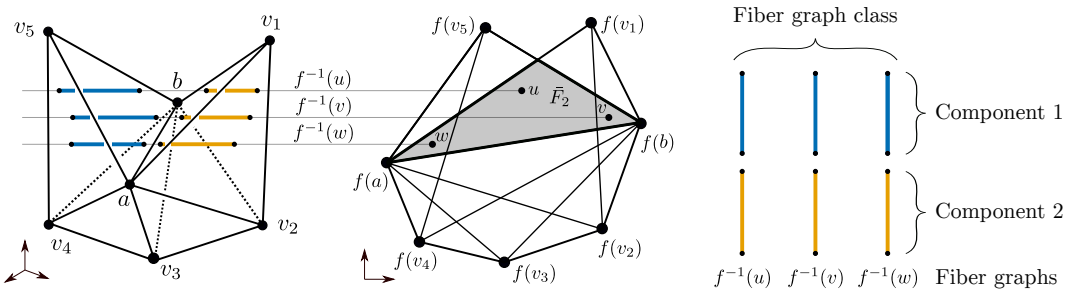


Figure 5 The fiber graph class of the face $\bar{F}_2$ is the collection of the fiber graphs of faces of the full arrangement that are in $\bar{F}_2$. All yellow (and respectively blue) fiber components correspond to one another; they change continuously from one to another as a fiber point moves around $\bar{F}_2$. All corresponding fiber graph components form a component of the fiber graph class of $\bar{F}_2$.

By reframing the computation of the Reeb space in terms of fiber graph classes, their components, and the singular correspondence graph $\bar{H}$, we can optimize running time and memory efficiency in several ways. First, we can completely skip the computation of all nonessential fiber graphs because the correspondence between two fiber graph class components can only be established with components of essential fiber graphs. Second, we can save a significant amount of memory, which is the primary bottleneck of the arrange and traverse algorithm, by representing every fiber graph class component as a single vertex in $\bar{H}$. Finally, since all essential faces are incident to singular segments, we do not have to compute the full arrangement A, only the singular one $\bar{A}$. Then we can identify the essential faces by finding the intersections between $\bar{A}$ and the regular segments.

3.2 Algorithm Description

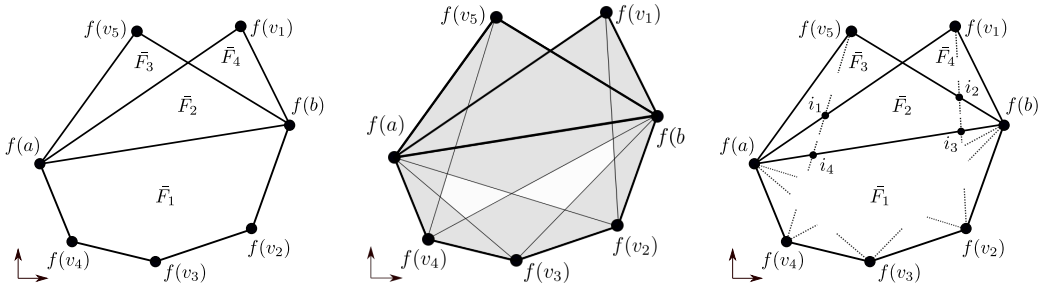
We describe the *singular arrange and traverse algorithm* in three stages. First, we compute all essential faces in the *geometric preprocessing stage*. Then, we compute their fiber graphs and the singular correspondence graph in the *topological traversal stage*. Finally, we compute the geometry of the sheets and their adjacency in the *geometric postprocessing stage*.

Stage I. Geometric Preprocessing

In the first stage, we compute all essential faces *implicitly* without using the full arrangement A. To do so, we compute the singular set S_f, then map it to the range to compute the singular arrangement $\bar{A}$, and finally we compute the intersection of $\bar{A}$ and the regular segments. We store all regular segments that intersect the boundary $\partial\bar{F}$ of each face $\bar{F}$ of $\bar{A}$ in a *circular counterclockwise ordered list* $Q_{\bar{F}}$.

When two regular segments intersect the interiors of two different half-edges or vertices of $\bar{F}$ we use the counterclockwise order defined on the boundary of $\bar{F}$ by the DCEL data structure of $\bar{A}$. When two regular segments intersect the interior of the same half-edge of $\bar{F}$, we compare the barycentric coordinates of the points of intersection. When two regular segments share a vertex of $\bar{F}$ as a common endpoint, we use their radial ordering. We use our running example to illustrate this in Figure 6, where $Q_{\bar{F}_2} = \{f(v_1v_2), f(v_1v_2), f(v_4v_5), f(v_4v_5)\}$.

Each essential face $F \subseteq \bar{F}$ is implicitly represented with the part of its boundary that intersects $\partial\bar{F}$. This can be identified by pairs of consecutive regular segments in $Q_{\bar{F}}$. Note also that an essential face may be represented multiple times when $\partial F \cap \partial\bar{F}$ has multiple connected components. Such is the case for the essential face whose boundary vertices include i_1, i_2, i_3 and i_4 in the middle of face $\bar{F}_2$ in Figure 6c.



(a) Singular arrangement. **(b)** Essential faces (shaded). **(c)** Regular segment intersections.

Figure 6 Our algorithm computes the arrangement $\bar{A}$ of all singular segments (a), then idenfities all essential faces (b) by computing the intersections between $\bar{A}$ and all regular segments (c).

Stage II. Topological Traversal

In the second stage, we traverse the faces of the singular arrangement along their shared segments using breadth-first search (BFS) to compute the essential fiber graphs and the singular correspondence graph $\bar{H}$. Starting with the unbounded face outside $\bar{A}$, whose fiber graph is empty, we visit one of its incident bounded faces $\bar{F}$ via a segment $f(ab)$, where ab is an edge in K. The edge ab is definite (see Section 2) because $f(ab)$ is on the boundary of $\bar{A}$, so we add the triangles abv, where v is in the upper (or lower, depending on the direction of travel) link of ab, to the first essential fiber graph in the fiber graph class of $\bar{F}$.

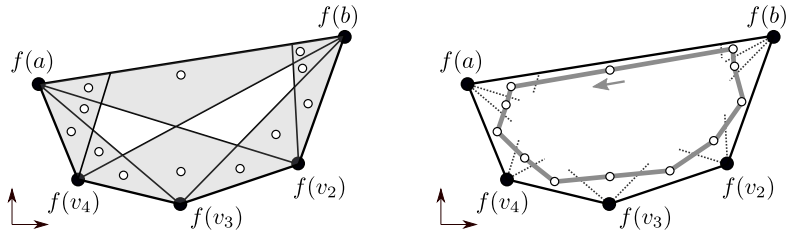
Next, we compute all other essential fiber graphs of $\bar{F}$ with a loop operation. A *loop operation* implicitly loops around the inner boundary of $\bar{F}$ to visit all essential faces in $\bar{F}$ using the order $Q_{\bar{F}}$ we have established in the geometric preprocessing stage. During the loop of $\bar{F}$, we cross from one essential face to the next via the regular segments listed in $Q_{\bar{F}}$. We add and remove the appropriate triangles related to the upper and lower links of the edges that map to those segments (see Section 2.1).

After we have computed the essential faces of $\bar{F}$, we visit its neighboring faces. We compute the new fiber graphs by adding and removing triangles according to the upper and lower links of the singular edges that map to shared segments on the boundaries of incident singular faces. We continue to loop each face as we visit it in the BFS traversal until we have computed the fiber graphs of all essential faces. For an example of the loop operation, see Figure 7, and for an example of the overall traversal, see Figure 8.

Finally, we construct the singular correspondence graph $\bar{H}$. Let $\bar{F}$ and $\bar{F}'$ be two singular faces which are incident via the segment $f(ab)$, and let F and F' be any two essential faces in $\bar{F}$ and $\bar{F}'$ respectively, which are also incident via $f(ab)$. We use the upper and lower links of ab to determine which components in the fiber graphs of F and F' are not affected by the transition from F to F' via $f(ab)$ and establish correspondence between them [26]. This allows us to establish correspondence between the components of the fiber graph classes of $\bar{F}$ and $\bar{F}'$. We do this for all pairs of incident singular faces.

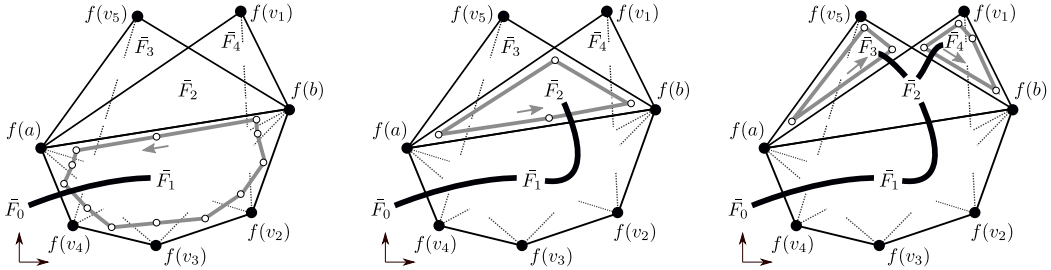
Stage III. Geometric Postprocessing

Once we have computed the singular correspondence graph, we can compute the geometry and adjacency of the sheets of the Reeb space R_f. By the geometry of a sheet of the Reeb space, we mean its image into the plane via $\pi : R_f \to \mathbb{R}^2$. Let L be a sheet that matches a connected component of $\bar{H}$. Then, the geometry of L is exactly the union of faces in the singular arrangement whose fiber graph class components span that component of



(a) Essential fiber graphs of $\bar{F}_1$. **(b)** Looping all essential fiber graphs.

Figure 7 Computing all essential fiber graphs of $\bar{F}_1$ from Figure 4 with a loop operation.



(a) Looping $\bar{F}_1$. **(b)** Looping $\bar{F}_2$. **(c)** Looping $\bar{F}_3$ & $\bar{F}_4$.

Figure 8 Traversal of the singular arrangement and looping each face.

$\bar{H}$. Sheets are adjacent when they have incident faces in the singular arrangement. The singular correspondence graph and the Reeb space of our running example are given in Figure 9. Notice how much smaller the singular correspondence graph is compared to the correspondence graph in Figure 3b.

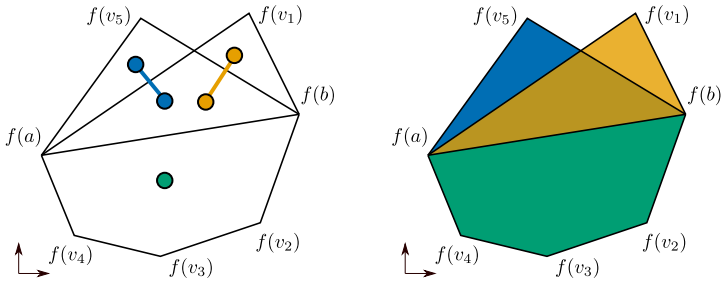
3.3 Nested Faces

We now address the case where the image of the singular set is *not* connected. In this case, either the unbounded face of the singular arrangement has more than one hole, or some bounded face has at least one hole, or both. If the unbounded face of the arrangement has multiple holes, we can traverse them independently. If a bounded face $\bar{F}_1$ has a nested face $\bar{F}_2$ (a hole), we find the shortest path in the 1-skeleton of K that connects a vertex that maps to $\partial\bar{F}_1$ to a vertex that maps to $\partial\bar{F}_2$. We mark the regular edges on this path as *pseudo-singular* and compute the arrangement of all singular and pseudo-singular segments, which is now guaranteed to be connected. After this, our algorithm proceeds as usual.

Note that this may introduce additional faces in the singular arrangement, such as $\bar{F}_3$ in Figure 10, whose nonessential fiber graphs will be computed by our algorithm. We will demonstrate that this is not an issue in practice and that only a negligible number of pseudo-singular edges are added for real-life datasets we evaluate in Section 6.

3.4 Higher Dimensional Domains

Finally, we describe an extension of our algorithm for computing the Reeb space of a PL map $f : |K| \to \mathbb{R}^2$, where K is a triangulation of a d-manifold with $d > 3$. Due to the skeleton lemma [18], the Reeb space of $f : |K| \to \mathbb{R}^2$ is homeomorphic to the Reeb space of $f' : K^{(3)} \to \mathbb{R}^2$, which is the restriction of f to the 3-skeleton of K. In order to compute the Reeb space of f', we compute all singular edges of K [16] and discard all simplices



(a) Correspondence graph. **(b)** Reeb space.

Figure 9 Singular correspondence graph and Reeb space of the toy example from Figure 1.

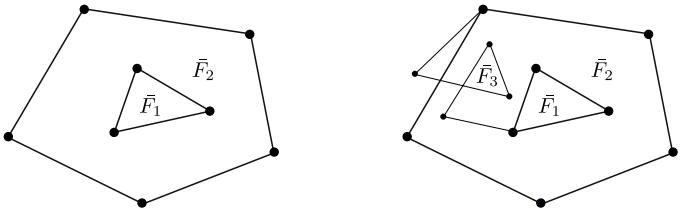


Figure 10 Connecting nested faces by adding pseudo-singular segments.

of dimension bigger than three from K. Then, we map the singular edges to the plane, compute their arrangement, and traverse it to compute the singular correspondence graph. The only difference is that we need a new subroutine to track the connectivity of fiber graphs. This is because the regular fibers of f' are not PL curves but the 1-skeleton of triangulated $(d-2)$-manifolds. This requires more involved data structures for dynamic graph connectivity tracking, similar to how Reeb graph algorithms handle the scalar case [15, 33].

4 Proofs

In this section we prove the correctness and worst-case running time of our algorithm using our definitions from the start of Section 3.

4.1 Correctness

First, we prove the correctness of each stage in our algorithm (see Section 3.2).

▶ **Lemma 5** (Essential Faces). *The singular arrange and traverse algorithm correctly identifies all essential faces.*

Proof. Let F be an essential face of the full arrangement A, and let $\bar{F}$ be the face of the singular arrangement $\bar{A}$ that contains F. The case when $F = \bar{F}$ is trivial because then F is already correctly represented in $\bar{A}$. Suppose that $F \subset \bar{F}$. Let B be a connected component of $\partial F \cap \partial \bar{F}$. Then B is a path from i_1 to i_2, where i_1 and i_2 are the intersection points of two regular segments s_1 and s_2 with $\partial \bar{F}$.

First, we examine the case where $B = i_1 = i_2$. In the DCEL of the full arrangement A, the half-edges of the boundary cycle of F that originate from s_1 and s_2 must be consecutive. This means that no other regular segment comes between s_1 and s_2 in the counterclockwise order of segments around i_1 in A. Otherwise, that other segment would be on the boundary of F. Therefore, the segments s_1 and s_2 are present as a consecutive pair in the circular list

$Q_{\bar{F}}$ of regular segments of $\bar{F}$. If on the other hand $i_1 \neq i_2$, then B is subdivided into at least one half-edge in the DCEL of A. No other regular segment intersects the interior of B, because then the essential face F would have been subdivided differently in A. Similarly to the previous case, other regular segments whose endpoint is either i_1 or i_2 are either before s_1 or after s_2 in the counterclockwise order of segments around those vertex points. Therefore, s_1 and s_2 are consecutive in $Q_{\bar{F}}$.

Conversely, we can use a similar argument to show that each consecutive pair of regular segments in $Q_{\bar{F}}$ can be uniquely identified with an essential face. ◀

▶ **Lemma 6** (Essential Fiber graphs). *The singular arrange and traverse algorithm correctly computes all essential fiber graphs.*

Proof. Since all essential faces are identified correctly by Lemma 5, our algorithm can traverse them, analogously to the arrange and traverse algorithm, but in a different order. To see why this does not affect the correctness, consider a path P in the range that only intersects regular or singular segments, from the unbounded outside face to an essential face. Any such path gives rise to a sequence of additions and removals of triangles, based on the upper and lower links of edges whose segments P intersects. This sequence can be used to build the fiber graph of the essential face, which is the same as the fiber graph computed by our algorithm. Additional pseudo-singular segments in $\bar{A}$ (see Section 3.3) do not affect the correctness because our algorithm still identifies such a path P correctly. ◀

▶ **Lemma 7** (Singular Correspondence Graph). *The connected components of the singular correspondence graph correctly represent the sheets of the Reeb space.*

Proof. The singular correspondence graph $\bar{H}$ can be obtained from the correspondence graph H by contracting all vertices that are associated with the components of fiber graphs within the same fiber graph class. A contraction of a graph does not affect its connectivity. Each vertex of $\bar{H}$ simply represents a larger area in the range of fiber points whose fibers have the same topology. The singular correspondence graph can be computed correctly by establishing correspondence only between the essential fiber graphs because nonessential fiber graphs do not introduce correspondences with new fiber graph classes. They only correspond directly to other components within their own class, and it is precisely these correspondences that are lost in the contraction of H to H'. The correctness of the correspondence procedure between fiber graphs is proven in the original arrange and traverse algorithm [26]. ◀

▶ **Corollary 8** (Reeb space). *The singular arrange and traverse algorithm correctly computes the Reeb space of a generic PL map $f : |K| \to \mathbb{R}^2$, where K is a triangulation of a 3-manifold.*

Proof. By Lemma 5 and Lemma 6, the essential faces and essential fiber graphs are computed correctly. By Lemma 7, the singular correspondence graph represents the sheets of the Reeb space and is computed correctly. Therefore, the singular arrange and traverse algorithm correctly computes the Reeb space of f. ◀

4.2 Complexity

Let N_e and N_t be the number of edges and triangles in K respectively. Let N_s be the number of singular edges. Let k_s be the number of intersections between all singular segments, and let k_r be the number of intersections between all regular and singular segments. First, we compute the arrangement of all singular segments in time $O((N_s + k_s) \log N_s)$. Next, we compute the number of intersections between all regular and singular segments and

their relative order in time $O((N_e + k_r) \log N_e)$. We use a red-blue segment intersection algorithm (see Section 2.2) and sort the intersection points along each singular segment using their barycentric coordinates. This concludes the geometric part of our computation. What remains is the complexity of computing the essential fiber graphs and the singular correspondence graph, which we demonstrate with the following lemma.

▶ **Lemma 9.** *Computing the essential fiber graphs and the singular correspondence graph has worst-case running time* $O(N_s N_t \log N_t)$.

Proof. Let $k(e)$ be the number of times an edge $e \in K$ is visited by the algorithm. A visit can mean either crossing a singular segment in the traversal from one face to another in the singular arrangement, or crossing a regular segment in the looping of a single face. When we visit an edge with degree $\deg(e)$, where we define the degree of an edge as the number of its incident triangles, we add and remove at most $\deg(e)$ triangles from the relevant fiber graphs, and perform at most $\deg(e)$ connectivity queries. Each of those operations takes at most $O(\log N_t)$ time if we represent each fiber graph with a balanced search tree [13].

The overall worst-case running time can then be obtained by summing the costs of all visits across all edges $O\big(\sum_{e \in E} k(e) \deg(e) \log(N_t)\big)$, where E is the set of edges in K. Since we do not consider any intersections between regular segments, each segment is intersected, and hence visited, at most N_s times. Using $k(e) \leq N_s$ we obtain $O\big(N_s \log(N_t) \sum_{e \in E} \deg(e)\big)$. Using a straightforward counting argument, we can show that $\sum_{e \in E} \deg(e) = 3N_t$. This leads to a worst-case running time of $O(N_s N_t \log N_t)$. ◀

Computing the connected components of $\bar{H}$ can be done in linear time in its size with a graph search. The geometry of the sheets of the Reeb space can be obtained by combining all faces of the singular arrangement that belong to each sheet. Thus, *the overall complexity of our algorithm is* $O(N_s N_t \log N_t)$. Note that, practical performance depends primarily on k_s and k_r, which determine how many times an edge is actually visited.

When $O(N_s) = O(N_e)$ our algorithm has the same running time as arrange and traverse. However, our algorithm is significantly more efficient for practical data sets where the number of singular edges is orders of magnitude less than the number of regular edges, as we will demonstrate in our evaluation in Section 6.

5 Implementation

We provide an open-source implementation of the singular arrange and traverse algorithm in C++. The input is a tetrahedral mesh with two scalar values defined at each vertex, represented in the `.vtu` file format with VTK [36]. For the geometric preprocessing operations we used CGAL 6.0.2 [39] with the exact predicates and exact constructions kernel [5]. We compute the red-blue intersections with CGAL's `box_intersection_d` [28], which has quadratic complexity. While, this is not optimal, it has an efficient implementation.

We track changes in the connectivity of fiber graphs with a graph search over the affected [26] fiber components. While this operation is not logarithmic in the size of the fiber, in practice most of the connectivity computations are done within the loop operation for a face of the singular arrangement, so all triangles can be directly added or removed from the same fiber graph component without affecting the connectivity.

Another important practical optimization, similar to arrange and traverse, is to only keep the fiber graphs at the wave front of the BFS search in memory since those are the only ones needed to establish correspondence to create the singular correspondence graph [26]. While

degenerate cases, which occur when the input PL map is not generic (see Section 2), can be handled robustly with simulation of simplicity [25], random numerical perturbation with double floating-point precision was sufficient for our tests.

6 Evaluation

In this section, we perform a practical evaluation of the efficiency of our algorithm in terms of running time and memory. We benchmark our algorithm against the arrange and traverse algorithm [26], since that is the only other open-source exact Reeb space implementation known to the authors. Our tests were performed on a PC with two Intel Xeon Gold 6130 processors with 32 cores clocked at 2.1GHz and 384GB memory with Rocky Linux 9.6. We measured execution time using the C++ `chrono` library and estimated memory usage via the process's resident set size by reading `/proc/self/status`.

We benchmark our algorithm on three numerical simulation datasets – MVK [8, 37], ethane-diol [7, 41] and enzo [6, 7]. MVK comes from a quantum chemistry simulation studying photo-induced changes in electron transition orbitals. Ethane-diol is a molecular dynamics simulation of a molecule's electron density field and its reduced gradient. Lastly, the origin of Enzo is cosmological simulation of the universe's expansion; here, the bivariate field represents matter and dark matter concentration. Each dataset is given on a three-dimensional regular grid and then tetrahedralized using Paraview's [1] default filter.

We summarize our results in Table 1. Columns labeled $\times$ show the speedup of our algorithm over arrange and traverse. In the time columns, this indicates how many times faster it is; in the memory columns, how much less memory it uses. The timings for the geometric preprocessing stage are given in the *Prep.* column and the timings for the topological traversal stage are given in the R_S column. The timings for the geometric postprocessing stage are omitted, as the computation time is negligible compared to the first two stages and depends on how the Reeb space will be used in downstream tasks. Each entry is rounded to the nearest whole number. We downsampled each dataset multiple times to obtain a wide range of data sizes and proportion of the number singular edges, ranging from 0.20% for MVK to 3.73% for ethane-diol and finally 43.78% for enzo. The enzo datasets and ethane-diol with 3,150K tetrahedra required the use of perturbation with strength 0.0001.

Our *primary result* is that the singular arrange and traverse algorithm is up to four orders of magnitude faster for MVK at 189K tetrahedra. For this dataset, arrange and traverse requires over 13 hours of computation time and about 100 GB of memory. Singular arrange and traverse not only takes less than 3 seconds and uses less than 700 MB of memory, but is also able to compute the Reeb space at full resolution (1,540K tetrahedra) in less than 10 seconds and 6 GB of memory. This speedup is due to the small number of singular edges, which our algorithm can successfully exploit.

Although the two other datasets have a larger singular set, we still see a significant speedup – at least two orders of magnitude for ethane-diol and at least one for enzo. Note that even though the number of singular edges in enzo is about half the total number of edges, we still see a significantly higher speedup – not just two-fold but up to 23x. This is due to the fact that the singular arrange and traverse algorithm not only allows us to avoid computing nonessential faces, but also allows us to group the computation of regular fiber graphs and significantly reduce the size of the correspondence graph.

In the case of MVK and ethane-diol, we are able to compute the Reeb space of their original non-downsampled resolution. This is something that would have been extremely difficult with arrange and traverse. The enzo dataset, however, has 83 million tetrahedra in its non-downsampled resolution and a high proportion of singular edges, which makes it difficult for our algorithm to handle.

Table 1 Performance metrics for real-life datasets. We report the number of input tetrahedra N_T, the percentage of singular edges N_s relative to all edges N_e, computation times, and memory usage for both preprocessing and Reeb space computations, including scaling factors (×) relative to arrange and traverse. Runs with over a day of compute were terminated and marked with "–".

	N_T	N_s/N_e	Time				Memory			
			Prep.(s)	×	R_S(s)	×	Prep.(Mb)	×	R_S(Mb)	×
MVK	22K	2.93%	2	14	<1	2K	<1	5K	<1	1K
	53K	1.70%	3	28	<1	6K	<2	12K	<1	4K
	189K	0.78%	8	72	2	25K	<2	64K	<1	18K
	1,540K	0.20%	41	–	6	–	<2	–	<1	–
Ethane-diol	25K	12.09%	85	0.9	30	129	204	58	237	23
	46K	9.33%	153	1.0	50	204	311	84	433	26
	113K	6.50%	402	1.4	123	426	600	139	810	46
	384K	3.73%	1,843	–	556	–	1,610	–	2,059	–
	3,150K	1.44%	19,259	–	5,238	–	7,438	–	7,389	–
Enzo	1K	49.88%	8	0.2	15	8	107	3	124	6
	4K	48.53%	189	0.1	323	13	869	4	988	17
	9K	44.62%	2,535	0.1	4,012	23	3,543	4	4,743	30
	14K	43.78%	7,127	–	11,294	–	7,882	–	11,140	–

We verified the correctness of our algorithm by comparing the sheets it outputs against arrange and traverse. For all datasets for which we could compute arrange and traverse, the outputs of both algorithms were identical. Finally, we note that the amount of additional computation needed to connect the nested faces was negligible. In the MVK datasets our algorithm added one to two additional pseudosingular edges to connect the arrangement and avoid nested faces. In the ethane-diol datasets that number was from 20 to 94.

The geometric preprocessing stage of our algorithm is significantly faster for MVK and ethane-diol and uses substantially less memory. Our memory usage improvement holds for enzo as well, but it takes up to 10 times longer to compute, even though arrange and traverse computes the arrangement of all segments. This may be addressed by implementing a more efficient algorithm for red-blue segment intersection.

7 Conclusion

In this paper, we presented the singular arrange and traverse algorithm for the exact and efficient computation of Reeb spaces of bivariate PL maps. This algorithm takes advantage of the fact that datasets with a small number of singular edges, relative to the total number of edges, have smaller Reeb spaces, which allows us to avoid nonessential computation from previous Reeb space algorithms such as arrange and traverse [26].

We presented a reformulation of the computation in terms of fiber graph classes and the singular correspondence graph, enabling the collective treatment of many related fiber graphs and thereby improving efficiency. We further provided an exact implementation of this approach and benchmarked it against the original arrange and traverse algorithm, demonstrating speedups ranging from one to four orders of magnitude across multiple real-world datasets derived from numerical simulations.

Future work will focus on optimising the running time through parallelisation as well as simplification for datasets with large singular sets. Other open problems include non-heuristic methods for dealing with nested faces and extensions to higher dimensional ranges.

References

1 James Ahrens, Berk Geveci, and Charles Law. Paraview: An end-user tool for large-data visualization. *Visualization Handbook*, pages 717–731, 2005. doi:10.1016/B978-012387582-2/50038-1.

2 Ivan J. Balaban. An optimal algorithm for finding segments intersections. In *Proceedings of the Eleventh Annual Symposium on Computational Geometry*, SCG '95, page 211, New York, NY, USA, 1995. Association for Computing Machinery. doi:10.1145/220279.220302.

3 Harsh Bhatia, Attila G. Gyulassy, Vincenzo Lordi, John E. Pask, Valerio Pascucci, and Peer-Timo Bremer. TopoMS: Comprehensive topological exploration for molecular and condensed-matter systems. *Journal of Computational Chemistry*, 39(16):936–952, 2018. doi:10.1002/jcc.25181.

4 S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1-3):5–22, February 2008. doi:10.1016/j.tcs.2007.10.018.

5 Hervé Brönnimann, Andreas Fabri, Geert-Jan Giezeman, Susan Hert, Michael Hoffmann, Lutz Kettner, Sylvain Pion, and Stefan Schirra. 2D and 3D linear geometry kernel. In *CGAL User and Reference Manual*. CGAL Editorial Board, 6.0 edition, 2024. URL: https://doc.cgal.org/6.0/Manual/packages.html#PkgKernel23.

6 Greg L. Bryan, Michael L. Norman, Brian W. O'Shea, Tom Abel, John H. Wise, Matthew J. Turk, Daniel R. Reynolds, David C. Collins, Peng Wang, Samuel W. Skillman, Britton Smith, Robert P. Harkness, James Bordner, Ji-hoon Kim, Michael Kuhlen, Hao Xu, Nathan Goldbaum, Cameron Hummels, Alexei G. Kritsuk, Elizabeth Tasker, Stephen Skory, Christine M. Simpson, Oliver Hahn, Jeffrey S. Oishi, Geoffrey C. So, Fen Zhao, Renyue Cen, Yuan Li, and The Enzo Collaboration. ENZO: An Adaptive Mesh Refinement Code For Astrophysics. *The Astrophysical Journal Supplement Series*, 211(2):19, March 2014. doi:10.1088/0067-0049/211/2/19.

7 Hamish Carr, Zhao Geng, Julien Tierny, Amit Chattopadhyay, and Aaron Knoll. Fiber surfaces: Generalizing isosurfaces to bivariate data. *Computer Graphics Forum*, 34(3):241–250, 2015. doi:10.1111/cgf.12636.

8 Pratip Chakraborty, Rafael C. Couto, and Nanna H. List. Deciphering Methylation Effects on S2($\pi\pi^*$) Internal Conversion in the Simplest Linear α,β-Unsaturated Carbonyl. *The Journal of Physical Chemistry A*, 127(25):5360–5373, 2023. doi:10.1021/acs.jpca.3c02582.

9 Amit Chattopadhyay, Hamish Carr, David Duke, and Zhao Geng. Extracting jacobi structures in reeb spaces. In N. Elmqvist, M. Hlawitschka, and J. Kennedy, editors, *EuroVis - Short Papers*. The Eurographics Association, 2014. doi:10.2312/eurovisshort.20141156.

10 Amit Chattopadhyay, Yashwanth Ramamurthi, and Osamu Saeki. An Algorithm for Fast and Correct Computation of Reeb Spaces for PL Bivariate Fields. doi:10.48550/arXiv.2403.06564.

11 Bernard Chazelle and Herbert Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM*, 39(1):1–54, January 1992. doi:10.1145/147508.147511.

12 James R. Clough, Nicholas Byrne, Ilkay Oksuz, Veronika A. Zimmer, Julia A. Schnabel, and Andrew P. King. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):8766–8778, 2022. doi:10.1109/TPAMI.2020.3013679.

13 Kree Cole-McLaughlin, Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Loops in reeb graphs of 2-manifolds. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, pages 344–350, San Diego California USA, June 2003. ACM. doi:10.1145/777792.777844.

14 Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer Berlin Heidelberg, 2008. doi:10.1007/978-3-540-77974-2.

15 Harish Doraiswamy and Vijay Natarajan. Efficient algorithms for computing Reeb graphs. *Computational Geometry*, 42(6-7):606–616, August 2009. doi:10.1016/j.comgeo.2008.12.003.

16 Herbert Edelsbrunner and John Harer. *Jacobi sets of multiple Morse functions*, volume 312, pages 37–57. Springer, 2004. doi:10.1017/CBO9781139106962.003.

17 Herbert Edelsbrunner and John Harer. *Computational Topology*. American Mathematical Society, Providence, Rhode Island, December 2009. doi:10.1090/mbk/069.

18 Herbert Edelsbrunner, John Harer, and Amit K. Patel. Reeb Spaces of Piecewise Linear Mappings. In *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry*, SCG '08, pages 242–250. ACM, 2008. doi:10.1145/1377676.1377720.

19 Andreas Fabri, Geert-Jan Giezeman, Lutz Kettner, Stefan Schirra, and Sven Schönherr. On the design of CGAL a computational geometry algorithms library. *Software: Practice and Experience*, 30(11):1167–1202, September 2000. doi:10.1002/1097-024X(200009)30:11<1167::AID-SPE337>3.0.CO;2-B.

20 Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors. *Handbook of Discrete and Computational Geometry*. Discrete Mathematics and Its Applications. CRC Press, third edition edition, 2018.

21 David Günther, Roberto A. Boto, Juila Contreras-Garcia, Jean-Philip Piquemal, and Julien Tierny. Characterizing molecular interactions in chemical systems. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2476–2485, 2014. doi:10.1109/TVCG.2014.2346403.

22 M. Hachani, A. Ouled Zaid, and W. Puech. Segmentation of 3D dynamic meshes based on Reeb graph approach. In *2014 22nd European Signal Processing Conference (EUSIPCO)*, pages 2175–2179, 2014.

23 C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. A Survey of Topology-based Methods in Visualization. *Computer Graphics Forum*, 35(3):643–667, 2016. doi:10.1111/cgf.12933.

24 Petar Hristov. Arrange_and_Traverse_Algorithm, 2025. GitHub repository. URL: https://github.com/peter-hristov/arrange-and-traverse-algorithm.

25 Petar Hristov, Ingrid Hotz, and Talha Bin Masood. Robust geometric predicates for bivariate computational topology. In *2025 IEEE Workshop on Topological Data Analysis and Visualization (TopoInVis)*, pages 63–73, 2025. doi:10.1109/TopoInVis68599.2025.00011.

26 Petar Hristov, Daisuke Sakurai, Hamish Carr, Ingrid Hotz, and Talha Bin Masood. Arrange and traverse algorithm for computation of reeb spaces of piecewise linear maps. *Computer Graphics Forum*, 44(5):e70206, 2025. doi:10.1111/cgf.70206.

27 Petar Georgiev Hristov. *Hypersweeps, Convective Clouds and Reeb Spaces*. PhD thesis, University of Leeds, June 2022. URL: https://etheses.whiterose.ac.uk/31965/.

28 Lutz Kettner, Andreas Meyer, and Afra Zomorodian. Intersecting sequences of dD iso-oriented boxes. In *CGAL User and Reference Manual*. CGAL Editorial Board, 6.0 edition, 2024. URL: https://doc.cgal.org/6.0/Manual/packages.html#PkgBoxIntersectionD.

29 Alexander Kuhn, Wito Engelke, Markus Flatken, Hans-Christian Hege, and Ingrid Hotz. Topology-Based Analysis for Multimodal Atmospheric Data of Volcano Eruptions. In Hamish Carr, Christoph Garth, and Tino Weinkauf, editors, *Topological Methods in Data Analysis and Visualization IV*, pages 35–50, Cham, 2017. Springer International Publishing.

30 Walter Motta, Jr Porto, Paulo, and Osamu Saeki. Stable Maps of 3-Manifolds Into the Plane and their Quotient Spaces. *Proceedings of the London Mathematical Society*, s3-71(1):158–174, July 1995. doi:10.1112/plms/s3-71.1.158.

31 Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. Topology of Deep Neural Networks. *Journal of Machine Learning Research*, 21(184):1–40, 2020. URL: https://jmlr.org/papers/v21/20-345.html.

32 Monica Nicolau, Arnold J. Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, April 2011. doi:10.1073/pnas.1102826108.

33 Salman Parsa. A deterministic $O(m \log m)$ time algorithm for the reeb graph. In *Proceedings of the Twenty-Eighth Annual Symposium on Computational Geometry*, SoCG '12, pages 269–276. Association for Computing Machinery, 2012. doi:10.1145/2261250.2261289.

34 Michael W. Reimann, Max Nolte, Martina Scolamiero, Katharine Turner, Rodrigo Perin, Giuseppe Chindemi, Paweł Dłotko, Ran Levi, Kathryn Hess, and Henry Markram. Cliques of Neurons Bound into Cavities Provide a Missing Link between Structure and Function. *Frontiers in Computational Neuroscience*, 11:48, June 2017. doi:10.3389/fncom.2017.00048.

35 Colin Patrick Rourke. *Introduction to Piecewise-Linear Topology*. Springer Study Edition Ser. Springer Berlin / Heidelberg, Berlin, Heidelberg, 1982.

36 Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit (4th ed.)*. Kitware, 2006.

37 Mohit Sharma, Talha Bin Masood, Nanna Holmgaard List, Ingrid Hotz, and Vijay Natarajan. Continuous scatterplot and image moments for time-varying bivariate field analysis of electronic structure evolution. *IEEE Transactions on Visualization and Computer Graphics*, 31(10):7229–7242, 2025. doi:10.1109/TVCG.2025.3543619.

38 T. Sousbie. The persistent cosmic web and its filamentary structure – I. Theory and implementation. *Monthly Notices of the Royal Astronomical Society*, 414(1):350–383, June 2011. doi:10.1111/j.1365-2966.2011.18394.x.

39 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 6.0 edition, 2024. URL: https://doc.cgal.org/6.0/Manual/packages.html.

40 J. Tierny and H. Carr. Jacobi Fiber Surfaces for Bivariate Reeb Space Computation. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):960–969, January 2017. doi:10.1109/TVCG.2016.2599017.

41 Julien Tierny, Guillaume Favelier, Joshua A. Levine, Charles Gueunet, and Michael Michaux. The topology toolkit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):832–842, 2018. doi:10.1109/TVCG.2017.2743938.

42 Xavier Tricoche and Christoph Garth. Topological Methods for Visualizing Vortical Flows. In Torsten Möller, Bernd Hamann, and Robert D. Russell, editors, *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, pages 89–107. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi:10.1007/B106657_5.

On Sparse Representations of 3-Manifolds

Kristóf Huszár
Institute of Geometry, Graz University of Technology, Austria

Clément Maria
Inria d'Université Côte d'Azur, Sophia Antipolis, France

Abstract

3-manifolds are commonly represented as triangulations, consisting of abstract tetrahedra whose triangular faces are identified in pairs. The combinatorial sparsity of a triangulation, as measured by the treewidth of its dual graph, plays a fundamental role in the design of parameterized algorithms. In this work, we investigate algorithmic procedures that transform or modify a given triangulation while controlling specific sparsity parameters. First, we revisit a standard, linear-time algorithm that converts a given triangulation into a Heegaard diagram of the underlying 3-manifold, showing that the construction preserves treewidth. We apply this construction to exhibit a fixed-parameter tractable framework for computing Kuperberg's quantum invariants of 3-manifolds. Second, we present a quasi-linear-time algorithm that retriangulates a given triangulation into one with maximum edge valence of at most nine, while only moderately increasing the treewidth of the dual graph. Combining these two algorithms yields a quasi-linear-time algorithm that produces, from a given triangulation, a Heegaard diagram in which every attaching curve intersects at most nine others.

2012 ACM Subject Classification Mathematics of computing → Geometric topology; Theory of computation → Fixed parameter tractability

Keywords and phrases computational 3-manifold topology, fixed-parameter tractability, Heegaard splittings and diagrams, triangulations, edge valence, treewidth, quantum invariants, tensor networks

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.58

Related Version *Full Version*: `https://arxiv.org/abs/2512.05779` [18]

Funding *Clément Maria*: Partially supported by the ANR project ANR-20-CE48-0007 (AlgoKnot).

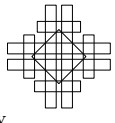
1 Introduction

Structural properties of triangulations – which are commonly used to encode 3-manifolds both in theory and in practice – can dramatically affect the feasibility of computations. Over the past decade, several fixed-parameter tractable (FPT) algorithms have been developed that efficiently solve provably hard problems on 3-manifolds, provided they are represented by triangulations that are sufficiently "thin" [5, 6, 7, 8, 9].[1] On input triangulations with bounded *treewidth*[2] these algorithms run in time polynomial in the number of tetrahedra.

Motivated by these algorithms, several recent papers have investigated the quantitative relationship between the treewidth (and other width parameters) of triangulations and the corresponding quantities in various related representations of 3-manifolds. It has been shown that 3-manifolds with small Heegaard genus [14, 19], as well as hyperbolic 3-manifolds with small volume [17, 30], admit triangulations with small treewidth. At the same time, for certain 3-manifolds, a large Heegaard genus or a "complicated" JSJ decomposition can entirely preclude the existence of triangulations with small treewidth [20, 21].[3]

[1] See [3, 27, 28, 29] for related FPT algorithms in knot theory, and the survey [25] for a broader context.
[2] Treewidth is a graph parameter that quantifies the similarity of a given graph to any tree. We define the treewidth of a triangulation $\mathcal{T}$ as the treewidth of its dual graph $\Gamma(\mathcal{T})$. See Section 2 for details.
[3] See [11, 26] for related structural results about knots and links and their diagrams.


© Kristóf Huszár and Clément Maria;
licensed under Creative Commons License CC-BY 4.0



In this work, we also focus on algorithmic transformations of 3-manifold triangulations with a view toward parameterized algorithms and sparsity. We provide two main contributions. First, we revisit and analyze a classical construction that turns a triangulation $\mathcal{T}$ of a closed 3-manifold $\mathcal{M}$ into a Heegaard splitting[4] and show that it yields a Heegaard diagram of $\mathcal{M}$, whose size and treewidth are linearly bounded by those of $\mathcal{T}$. More concretely, we prove

▶ **Theorem 1.** *Let $\mathcal{T}$ be a triangulation of a closed, orientable 3-manifold $\mathcal{M}$ with n tetrahedra and dual graph $\Gamma(\mathcal{T})$. Let $\mathcal{D} := \mathcal{D}(\mathcal{T})$ be the Heegaard diagram of $\mathcal{M}$ induced by $\mathcal{T}$. Then, for the number of vertices and for the treewidth[5] of $\mathcal{D}$ and $\Gamma(\mathcal{T})$ we have*

$$|V(\mathcal{D})| = 6n \quad \text{(1a)} \qquad \text{and} \qquad \operatorname{tw}(\mathcal{D}) \leq 12\operatorname{tw}(\Gamma(\mathcal{T})) + 11. \quad \text{(1b)}$$

Moreover, given $\mathcal{T}$, the induced Heegaard diagram $\mathcal{D}$ can be constructed in $O(n)$ time.

Guided by this result, in Section 5 we investigate the complexity of computing Kuperberg's quantum invariants of oriented 3-manifolds [23], which are obtained from a Heegaard diagram by evaluating an associated tensor network. First, we show that this construction is width-preserving (Lemma 18). This insight, combined with Theorem 1 and results on the complexity of evaluating tensor networks [31, 34] (cf. Theorem 19), provides a framework for computing Kuperberg's invariants from triangulations that is FPT in the treewidth (Theorem 15).

▶ Remark. The converse of Theorem 1, i.e., building a triangulation from a given Heegaard splitting in a width-preserving way, has been studied in [19] and, very recently, in [14], where an explicit algorithm for constructing a triangulation from a Heegaard diagram is provided.

Our second contribution investigates the interplay between the *maximum edge valence* and the treewidth of a 3-manifold triangulation $\mathcal{T}$. Analogous to the notion of vertex degree in graphs, the *valence of an edge* in $\mathcal{T}$ is defined as the number of tetrahedra in $\mathcal{T}$ that contain it. While treewidth can be viewed as a measure of *global* sparsity, the maximum edge valence captures *local* sparsity. The maximum edge valence of a triangulation can also provide valuable insights into geometric properties of the underlying 3-manifold [13, Sec. 3.6]. Having a triangulation $\mathcal{T}$ with small maximum edge valence can be computationally advantageous even when $\mathcal{T}$ already has small treewidth; see the full version [18] for a discussion.

Every closed orientable 3-manifold has a triangulation with maximum edge valence at most six [2]. Frick has shown by an explicit construction that any triangulation $\mathcal{T}$ can be modified into a triangulation $\mathcal{T}^\dagger$ of the same manifold with edge valences at most nine [13, Theorem 3.36]. However, this construction may significantly increase the treewidth of the resulting triangulation, because if $\mathcal{T}$ contains edges with large valence, the retriangulation procedure introduces large grid-like structures in $\mathcal{T}^\dagger$. In Section 4, we show how to circumvent this and obtain a different retriangulation $\mathcal{T}^*$ that also achieves $\Delta(\mathcal{T}^*) \leq 9$, while controlling the increase in treewidth by a polylogarithmic factor of the maximum edge valence of $\mathcal{T}$.

▶ **Theorem 2.** *There is an algorithm which, given a triangulation $\mathcal{T}$ of a closed 3-manifold $\mathcal{M}$ with n tetrahedra, $\mathbf{v}$ vertices, $\operatorname{tw}(\Gamma(\mathcal{T})) = \mathbf{tw}$, and maximum edge valence $\Delta(\mathcal{T}) = \boldsymbol{\Delta}$, constructs a triangulation $\mathcal{T}^*$ of $\mathcal{M}$ with $O((n+\mathbf{v}) \cdot \operatorname{poly}(\log_2(\boldsymbol{\Delta})))$ vertices and tetrahedra, $\operatorname{tw}(\Gamma(\mathcal{T}^*)) \leq \mathbf{tw} \cdot \operatorname{poly}(\log_2(\boldsymbol{\Delta}))$, and $\Delta(\mathcal{T}^*) \leq 9$. The algorithm runs in $O(n \log\log n)$ time.*

[4] A Heegaard splitting is a decomposition of a 3-manifold into two identical handlebodies, cf. Section 2.2.

[5] The number $V(\mathcal{D})$ of vertices and the treewidth $\operatorname{tw}(\mathcal{D})$ of a Heegaard diagram $\mathcal{D} = (\mathcal{S}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ are, by definition, respectively equal to those of its underlying graph, which is the 4-regular multigraph geometrically obtained by taking the union of the α- and β-curves and placing a vertex at each crossing.

Finally, the execution of the algorithms in Theorem 2 and Theorem 1 (in this order) yields the following corollary, which we believe to be of independent interest.

▶ **Corollary 3.** *There is a quasi-linear time algorithm that, given a triangulation $\mathcal{T}$ of a closed, orientable 3-manifold $\mathcal{M}$ with n tetrahedra, $\mathrm{tw}(\Gamma(\mathcal{T})) = \mathbf{tw}$, and maximum edge valence $\Delta(\mathcal{T}) = \boldsymbol{\Delta}$, constructs a Heegaard diagram $\mathcal{D} = (\mathcal{S}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ of $\mathcal{M}$ of size and genus $O(n\,\mathrm{poly}(\log(\boldsymbol{\Delta})))$, treewidth $O(\mathbf{tw} \cdot \mathrm{poly}(\log(\boldsymbol{\Delta}))$, where any α-curve intersects at most 9 β-curves, and any β-curve intersects at most 3 α-curves.*

2 Preliminaries

2.1 Graphs and their treewidth

By a *graph* we generally mean a *multigraph* $G = (V, E)$ with a finite set $V = V(G)$ of *vertices* and of a multiset $E = E(G)$ of two-element submultisets of V called *edges*. A *loop* is an edge of the form $\{v, v\}$, and a *multiedge* is an edge with multiplicity larger than one. A graph without loops or multiedges is *simple*. The *degree* $\deg(v)$ of $v \in V$ is the number of edges containing v, counted with multiplicity. A graph is *k-regular* if $\deg(v) = k$ for all $v \in V$.

Introduced in [36] (cf. [12]), the *treewidth* of a graph measures its similarity to any tree. A *tree decomposition* $\mathcal{X}$ of a graph $G = (V, E)$ is a pair $\mathcal{X} = (T, \mathcal{B})$ of a tree $T = (I, F)$ and a collection $\mathcal{B} = \{B_i : i \in I\}$ of *bag*s, where $B_i \subseteq V$ and the following three properties hold: **1.** $\bigcup_{i\in I} B_i = V$ (vertex coverage), **2.** for every $\{u, v\} \in E$ there exists $i \in I$ with $\{u, v\} \subseteq B_i$ (edge coverage), and **3.** for every $v \in V$, the set $T_v = \{i \in I : v \in B_i\}$ spans a connected subtree of T (subtree property). The *width* of the tree decomposition $\mathcal{X}$ equals $\max_{i\in I} |B_i| - 1$, and the *treewidth* $\mathrm{tw}(G)$ is defined as the smallest width of any tree decomposition of G.

2.2 3-Manifolds and their representations

A *d-dimensional topological manifold with boundary* (hereafter *d-manifold*) $\mathcal{M}$ is a topological space[6] locally homeomorphic to $\mathbb{R}^d$, i.e., every point of $\mathcal{M}$ has a neighborhood homeomorphic to $\mathbb{R}^d$ or to the upper half-space $\{(x_1, \ldots, x_d) \in \mathbb{R}^d : x_d \geq 0\}$. The *boundary* $\partial\mathcal{M}$ of $\mathcal{M}$ consists of all points of $\mathcal{M}$ that have no open neighborhood homeomorphic to $\mathbb{R}^d$. In this paper we only consider compact, orientable d-manifolds with $d \leq 3$. A manifold $\mathcal{M}$ is *closed* if it is compact and $\partial\mathcal{M} = \emptyset$. A *surface* is just a 2-manifold. We consider manifolds up to homeomorphism. See [39] for general background on 3-manifolds.

Triangulations and their dual graphs. As is common in algorithmic topology, we define a (3-dimensional) *triangulation* as a quotient space $\mathcal{T} = \Sigma/\Phi$, where $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ is a set of labeled and vertex-labeled abstract tetrahedra, and $\Phi = \{\varphi_1, \ldots, \varphi_m\}$ is a set of simplicial isomorphisms, called *gluing maps*, each of which identifies two triangles of the simplices in Σ.[7] For $i \in \{0, 1, 2, 3\}$, let $\mathcal{T}(i)$ denote the set of i-dimensional faces of the resulting quotient $\mathcal{T}$, and let $\mathcal{T}^{(i)}$, its *i-skeleton*, be $\mathcal{T}^{(i)} := \cup_{j=0}^{i} \mathcal{T}(j)$. The *valence* $\mathrm{val}(e)$ of an edge $e \in \mathcal{T}(1)$ is the number of tetrahedra in $\mathcal{T}$ that contain it and $\Delta(\mathcal{T}) := \max_{e\in\mathcal{T}(1)} \mathrm{val}(e)$. $\mathcal{T}$ *triangulates* the manifold $\mathcal{M}$ if the geometric realization of $\mathcal{T}$ is homeomorphic to $\mathcal{M}$. Triangulations of surfaces are defined analogously. Every surface and 3-manifold has a triangulation [33, 35].

[6] Some technical requirements about the space are intentionally omitted here, cf. [39, Definition 1.1.1].

[7] Each triangle is identified with at most one other triangle. We allow two pairs of triangles to be glued between the same pair of tetrahedra, as well as gluing two triangles within a single tetrahedron. Hence the triangulations considered here are not simplicial complexes but *regular simplicial cell complexes*. Nevertheless, the second barycentric subdivision $\mathcal{T}''$ of any triangulation $\mathcal{T}$ is always a simplicial complex.

Given a triangulation $\mathcal{T} = \Sigma/\Phi$, its *dual graph* (or *face-pairing graph*) $\Gamma(\mathcal{T})$ is the graph with $V(\Gamma(\mathcal{T})) = \Sigma$, and $\{\sigma_1, \sigma_2\} \in E(\Gamma(\mathcal{T}))$ if and only if there is a gluing map $\varphi \in \Phi$ that identifies a triangle of σ_1 with a triangle of σ_2 (where σ_1 and σ_2 may coincide). See Figure 1. We define the *treewidth of a triangulation* $\mathcal{T}$ as the treewidth of its dual graph $\Gamma(\mathcal{T})$.

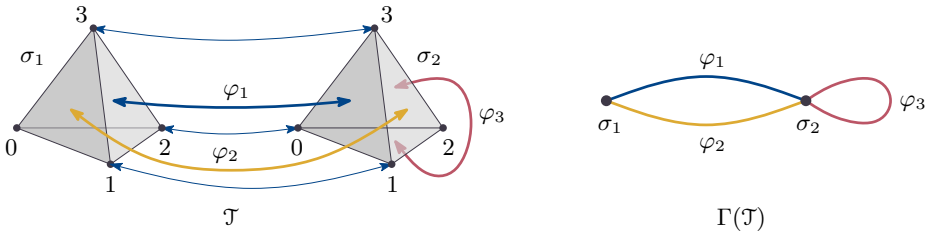


Figure 1 A triangulation $\mathcal{T} = \Sigma/\Phi$ with tetrahedra $\Sigma = \{\sigma_1, \sigma_2\}$, gluing maps $\Phi = \{\varphi_1, \varphi_2, \varphi_3\}$ and dual graph $\Gamma(\mathcal{T})$. φ_1 is explicitly defined as $\sigma_1(123) \leftrightarrow \sigma_2(103)$. Based on [20, Figure 1].

Handlebodies and Heegaard splittings. A *handlebody* $\mathcal{H}$ is a compact 3-manifold homeomorphic to a regular neighborhood of a bouquet of circles embedded in $\mathbb{R}^3$. Every closed, orientable 3-manifold $\mathcal{M}$ admits a *Heegaard splitting* [15, 16], i.e., a decomposition into two homeomorphic handlebodies $\mathcal{H}_1$ and $\mathcal{H}_2$ with a shared boundary. That is, $\mathcal{M} = \mathcal{H}_1 \cup \mathcal{H}_2$ and $\mathcal{H}_1 \cap \mathcal{H}_2 = \partial\mathcal{H}_1 = \partial\mathcal{H}_2 = \mathcal{S}$. The surface $\mathcal{S}$ is called a *splitting surface*, and its genus $g(\mathcal{S})$ is referred to as the *genus of the Heegaard splitting*. The minimum genus of any Heegaard splitting of $\mathcal{M}$ is called the *Heegaard genus* of $\mathcal{M}$. We refer to [38] for an extensive survey.

Heegaard diagrams. Heegaard diagrams were developed alongside Heegaard splittings to describe them. They also provide the basis for computing Kuperberg's invariants (Section 5). We follow the definitions of [23, Section 4]. In particular, we consider Heegaard diagrams in a sense more flexible than usual. A *handlebody diagram* is an ordered pair $(\mathcal{S}, \boldsymbol{\gamma})$ of a closed orientable surface $\mathcal{S}$ and a collection $\boldsymbol{\gamma} = \{\gamma_1, \ldots, \gamma_\ell\}$ of pairwise disjoint simple closed curves embedded in $\mathcal{S}$ that divide $\mathcal{S}$ into planar regions.[8] (Figure 2). A handlebody diagram $(\mathcal{S}, \boldsymbol{\gamma})$

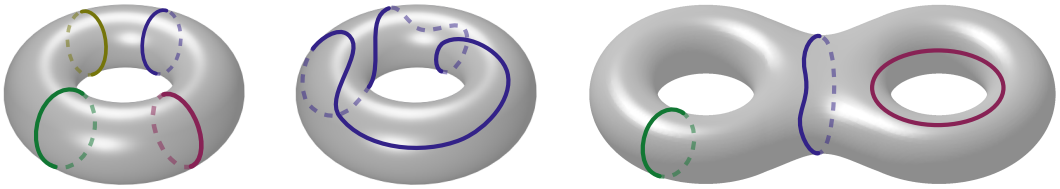


Figure 2 Illustrations of some handlebody diagrams. The surfaces are meant to be hollow.

can be seen as a prescription for gluing a handlebody to one side of $\mathcal{S}$. Specifically, consider the thickening $\mathcal{S} \times [0, 1]$, and for each $i \in \{1, \ldots, \ell\}$ attach to its *upper boundary* $\mathcal{S} \times \{1\}$ a thickened disk (called a 2-*handle*) along the curve γ_i.[9] Because of the assumption about $\boldsymbol{\gamma}$, each resulting new boundary component is a 2-sphere, each of which we fill in with a 3-ball. (Importantly, these fillings are unique up to isotopy; see [39, Lemma 2.5.3].) The union of the attached 2-handles and the filling 3-balls is a handlebody $\mathcal{H}$, which we have just glued to $\mathcal{S}$.

[8] Because of this requirement the number ℓ of curves in $\boldsymbol{\gamma}$ is at least the genus $g(\mathcal{S})$ of the surface $\mathcal{S}$, and $\ell = g(\mathcal{S})$ is commonly assumed in the literature. However, we will generally have $\ell > g(\mathcal{S})$.

[9] More precisely, we attach the 2-handle along the copy of γ_i in $\mathcal{S} \times \{1\}$.

With that, a *Heegaard diagram* is a triple $\mathcal{D} = (\mathcal{S}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ such that $(\mathcal{S}, \boldsymbol{\alpha})$ and $(\mathcal{S}, \boldsymbol{\beta})$ are handlebody diagrams relative to different sides of $\mathcal{S}$. That is, the 2-handles corresponding to $\boldsymbol{\alpha}$ (resp. $\boldsymbol{\beta}$) are attached to the upper boundary $\mathcal{S} \times \{1\}$ (resp. the *lower boundary* $\mathcal{S} \times \{0\}$) of the thickening $\mathcal{S} \times [0, 1]$. We refer to the curves in $\boldsymbol{\alpha}$ (resp. $\boldsymbol{\beta}$) as *α-curves* (resp. *β-curves*) and assign the same color to each family in illustrations; see Figure 3. A Heegaard diagram $\mathcal{D}$ can naturally be viewed as a 4-regular, surface-embedded graph, obtained geometrically by taking the union of the α- and β-curves and placing a vertex at each crossing.

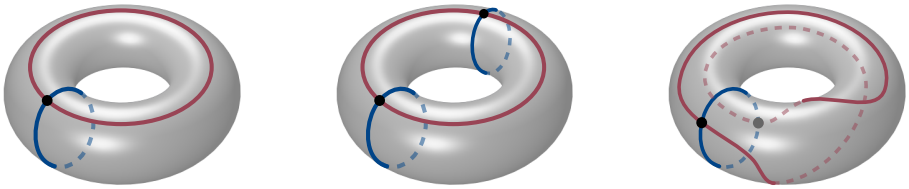
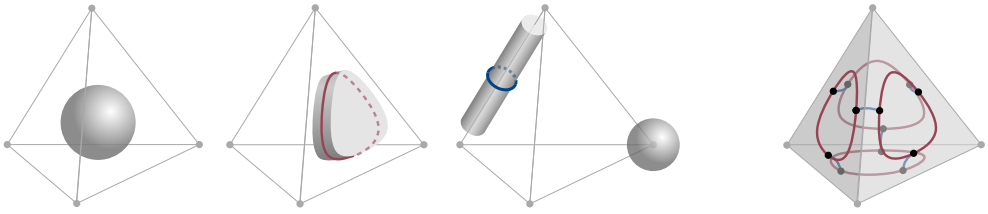


Figure 3 Two Heegaard diagrams of the 3-sphere, and one of the real projective 3-space $\mathbb{RP}^3$.

3 From triangulations to Heegaard diagrams

In this section, we revisit a classical construction that converts a triangulation $\mathcal{T}$ into a corresponding Heegaard diagram $\mathcal{D} = (\mathcal{S}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. We prove that this construction preserves both size and width up to a constant factor, as stated in Theorem 1. Moreover, using standard data structures, the conversion $\mathcal{T} \rightsquigarrow \mathcal{D}$ can be performed in time linear in the size of $\mathcal{T}$. We first review the construction via handles, then prove the inequalities (1a) and (1b) in Theorem 1, and finally discuss the algorithmic aspects of the construction.

Handle decompositions. Any triangulation $\mathcal{T}$ of a 3-manifold $\mathcal{M}$ induces a *canonical handle decomposition* $\mathrm{chd}(\mathcal{T})$ of $\mathcal{M}$, in which the *k-handles* (all 3-balls) correspond to the $(3-k)$-simplices of $\mathcal{T}$. Informally, $\mathrm{chd}(\mathcal{T})$ is the "thickening" of $\mathcal{T}$ (see Figure 4a). For each tetrahedron $\sigma \in \mathcal{T}(3)$, take a 0-handle centered at its barycenter. For each triangle $t \in \mathcal{T}(2)$, attach a 1-handle along a pair of disjoint disks connecting the 0-handles of the tetrahedra incident to t. For each edge $e \in \mathcal{T}(1)$, attach a 2-handle along an annulus to the union of the 0- and 1-handles corresponding to the tetrahedra and triangles incident to e. Finally, for each vertex $v \in \mathcal{T}(0)$, attach a 3-handle along a 2-sphere to fill the void surrounding v.



(a) The building blocks of the canonical handle decomposition $\mathrm{chd}(\mathcal{T})$. Meridian curves of the 1- and the 2-handle are also drawn.

(b) The α- and β-curves of $\mathcal{D}(\mathcal{T})$ restricted to a tetrahedron σ of $\mathcal{T}$.

Figure 4 The canonical handle decomposition and the induced Heegaard diagram.

The Heegaard diagram induced by $\mathcal{T}$. The union $\mathcal{H}_1$ of the 0- and 1-handles and the union $\mathcal{H}_2$ of the 2- and 3-handles of $\mathrm{chd}(\mathcal{T})$ are both handlebodies that together form a Heegaard splitting of the 3-manifold triangulated by $\mathcal{T}$. Letting $\mathcal{S} := \mathcal{H}_1 \cap \mathcal{H}_2$ and taking the meridional curves of the 2-handles and those of the 1-handles as α- and β-curves, respectively (Figure 4a), we obtain the Heegaard diagram $\mathcal{D}(\mathcal{T}) = (\mathcal{S}, \boldsymbol{\alpha}, \boldsymbol{\beta})$, where $\boldsymbol{\alpha} = \{\alpha_e : e \in \mathcal{T}(1)\}$ and $\boldsymbol{\beta} = \{\beta_t : t \in \mathcal{T}(2)\}$. We call $\mathcal{D}(\mathcal{T})$ the *Heegaard diagram induced by the triangulation* $\mathcal{T}$.

Proof of inequalities (1a) and (1b) in Theorem 1. For (1a), note that each triangle of $\mathcal{T}$ contains exactly three vertices of the Heegaard diagram $\mathcal{D} := \mathcal{D}(\mathcal{T})$ induced by $\mathcal{T}$ (Figure 4b), and each tetrahedron has four triangular faces identified in pairs, hence $|V(\mathcal{D})| = 3 \cdot 4n/2 = 6n$. To show (1b), take an optimal tree decomposition $\mathcal{X} = (T, \mathcal{B})$ of $\Gamma(\mathcal{T})$, where $T = (I, F)$ is a tree, $\mathcal{B} = \{B_i : i \in I\}$ is the corresponding set of bags, and $\max\{|B_i| : i \in I\} = \mathrm{tw}(\Gamma(\mathcal{T})) + 1$. We turn $\mathcal{X}$ into a tree decomposition $\mathcal{X}' = (T', \mathcal{B}')$ of $\mathcal{D}$ as follows.

- $T' = T$, i.e., the tree structure of the decomposition remains the same.
- $\mathcal{B}' = \{B'_i : i \in I\}$ and for each $i \in I$, the bag B'_i is obtained from B_i by replacing each $\sigma \in B_i$ with the (at most) 12 vertices of $\mathcal{D}$ incident to σ, see Figure 4b.

Clearly, $|B'_i| \leq 12\,|B_i|$. We now check that $\mathcal{X}' = (T', \mathcal{B}')$ is a tree decomposition of $\mathcal{D}$.

Vertex coverage. Consider a vertex $v \in V(\mathcal{D})$. By construction, there exists a tetrahedron $\sigma \in \mathcal{T}$, such that v is incident to σ. Thus, for any $i \in I$, $v \in B'_i \Leftrightarrow \sigma \in B_i$. However, since $\bigcup_{i \in I} B_i = V(\Gamma(\mathcal{T}))$, this implies $\bigcup_{i \in I} B'_i = V(\mathcal{D})$ as well.

Edge coverage. Let $e = \{u, v\} \in E(\mathcal{D})$ be any arc of the diagram. Again, by construction of $\mathcal{D}$, there exists a tetrahedron $\sigma \in \mathcal{T}$ that entirely contains e. Let B_i be a bag that contains σ. Then B'_i must contain both u and v.

Subtree property. Fix a vertex $v \in V(\mathcal{D})$ and consider $I'_v := \{i \in I : v \in B'_i\}$. We need to show that I'_v spans a connected subtree of $T' = T$. Let $\sigma, \tau \in \mathcal{T}$ be the tetrahedra containing v on their common triangular face (note that σ and τ may coincide) and define $I_\sigma := \{i \in I : \sigma \in B_i\}$ and $I_\tau := \{i \in I : \tau \in B_i\}$. From the definition of $\mathcal{B}'$ it follows that $I'_v = I_\sigma \cup I_\tau$. Since $\mathcal{X}$ is a tree decomposition of $\Gamma(\mathcal{T})$, each of I_σ and I_τ spans a connected subtree in T'. Moreover, $I_\sigma \cap I_\tau \neq \emptyset$: either $\sigma = \tau$, or $\{\sigma, \tau\} \in E(\Gamma(\mathcal{T}))$, so there exists a bag B_j containing both σ and τ. Thus, the connected subtrees spanned by I_σ and I_τ overlap, hence their union I'_v spans a connected subtree as well. ◀

Algorithmic aspects of Theorem 1. We briefly comment on the computational cost of extracting the Heegaard diagram $\mathcal{D}(\mathcal{T})$ from a given triangulation $\mathcal{T}$ with n tetrahedra. Recall that for an *abstract simplicial complex* $\mathcal{C}$, its barycentric subdivision $\mathcal{C}'$ is the simplicial complex whose ground set is $\mathcal{C}$, and a collection $\{\tau_0, \ldots, \tau_k\}$ forms a k-simplex in $\mathcal{C}'$ precisely when the simplices $\tau_i \in \mathcal{C}$ are nested, that is, when $\tau_0 \subset \cdots \subset \tau_k$. Although a triangulation $\mathcal{T}$ is generally not a simplicial complex, we can still form its barycentric subdivision by first subdividing each tetrahedron and then applying the gluing maps to identify the subdivided triangular faces. Performed twice, the resulting complex $\mathcal{T}''$ naturally contains both the canonical handle decomposition $\mathrm{chd}(\mathcal{T})$ and the induced Heegaard diagram $\mathcal{D}(\mathcal{T})$. As the barycentric subdivision of a tetrahedron contains 24 tetrahedra, $|\mathcal{T}''| = 24^2 |\mathcal{T}| = 576 |\mathcal{T}|$.

Extracting $\mathrm{chd}(\mathcal{T})$ from $\mathcal{T}''$. Every vertex v of $\mathcal{T}'$ is of the form $v = \{\tau\}$ for some simplex $\tau \in \mathcal{T}$. Let $\iota \colon \mathcal{T}'(0) \hookrightarrow \mathcal{T}''(0)$ be the natural inclusion defined by $\iota(\{\tau\}) = \{\{\tau\}\}$. For every $w \in \mathrm{im}(\iota)$, let h_w be the smallest subcomplex of $\mathcal{T}''$ containing all 3-faces incident to w. If w corresponds to a simplex $\sigma \in \mathcal{T}$ of dimension $3 - k$, then h_w is regarded as a k-handle. The collection $\{h_w : w \in \mathrm{im}(\iota)\}$ forms the canonical handle decomposition $\mathrm{chd}(\mathcal{T})$. See Figure 5.

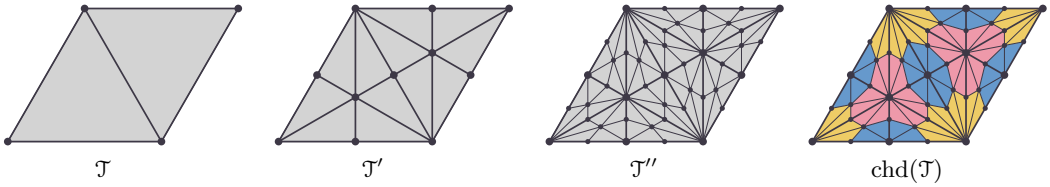


Figure 5 A 2-dimensional triangulation $\mathcal{T}$ with two triangles, its first and second barycentric subdivisions $\mathcal{T}'$ and $\mathcal{T}''$ (regarded geometrically), and the canonical handle decomposition chd($\mathcal{T}$).

Extracting $\mathcal{D}(\mathcal{T})$ from $\mathcal{T}''$. To extract the induced Heegaard diagram $\mathcal{D}(\mathcal{T})$ from $\mathcal{T}''$, we need to pinpoint its α- and β-curves, as well as the splitting surface $\mathcal{S}$ in $\mathcal{T}''$. We illustrate this via concrete examples; see Figure 6. We start with the α-curves. Let $e \in \mathcal{T}(1)$ be an edge of $\mathcal{T}$, and assume that e is contained in four distinct tetrahedra $\sigma_1, \sigma_2.\sigma_3, \sigma_4$ of $\mathcal{T}$. Let t_{ij} denote the triangular face shared by σ_i and σ_j. Consider the subcomplex C_e of $\mathcal{T}'$ spanned by the nine vertices $\{e\}, \{\sigma_1\}, \{\sigma_2\}, \{\sigma_3\}, \{\sigma_4\}, \{t_{12}\}, \{t_{23}\}, \{t_{34}\}$, and $\{t_{14}\}$. This complex C_e triangulates the *dual 2-cell transverse to* e (Figure 6, top left). Taking the second barycentric subdivision $\mathcal{T}''$ (Figure 6, top middle), we see that the α-curve α_e corresponding to e is readily obtained as the polygonal curve spanned by the vertices of C'_e adjacent to $\{\{e\}\}$ (Figure 6, top right). The β-curves can be located analogously (Figure 6, bottom row).

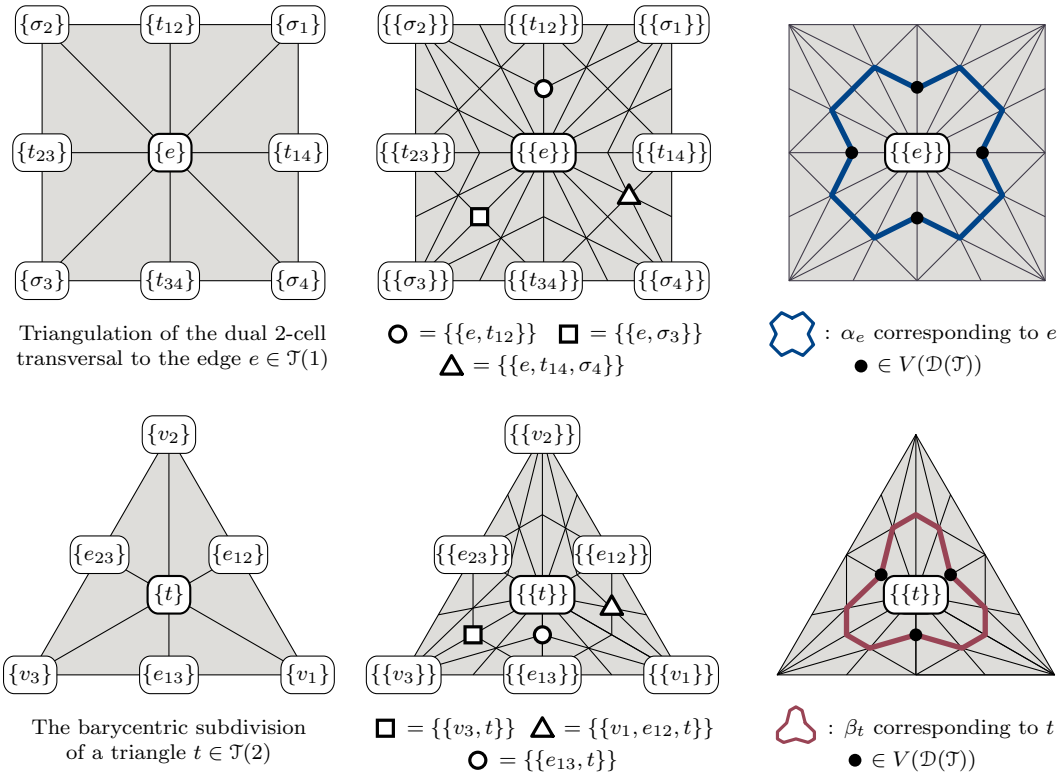


Figure 6 Locating the α- and β-curves $\mathcal{D}(\mathcal{T})$ in the second barycentric subdivision $\mathcal{T}''$ of $\mathcal{T}$.

A triangulation of the splitting surface $\mathcal{S}$ can also be readily obtained as the boundary of the subcomplex of $\mathcal{T}''$ formed by the union of the triangulated 2- and 3-handles of chd($\mathcal{T}$). Using standard data structures to represent these objects (such as in `Regina` [4]), all of these operations can be performed in linear time in terms of $n = |\mathcal{T}|$.

▶ Remark 4. Often it is assumed – and in certain applications even required, cf. [24] – that a Heegaard diagram $\mathcal{D} = (\mathcal{S}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ be *minimal*, i.e. $|\boldsymbol{\alpha}| = |\boldsymbol{\beta}| = g(\mathcal{S})$. In our setting, we can ensure this by first choosing arbitrary spanning trees T_1 and T_2 for the 1-skeleton $\mathcal{T}^{(1)}$ and the dual graph $\Gamma(\mathcal{T})$ of $\mathcal{T}$, respectively, and then retaining only those α-curves (resp. β-curves) that correspond to edges of $\mathcal{T}^{(1)}$ (resp. $\Gamma(\mathcal{T})$) *not* contained in T_1 (resp. T_2).

4 Retriangulating with constant edge valence and controlled treewidth

In this section, we describe and analyze a quasi-linear time procedure that retriangulates any 3-manifold triangulation $\mathcal{T}$ into a triangulation $\mathcal{T}^*$ with maximum edge valence at most 9 and treewidth increased only by a polylogarithmic factor of the original maximum edge valence.

▶ **Theorem 5** (Theorem 2; with explicit polylog-factors)**.** *There is an algorithm which, given a triangulation $\mathcal{T}$ of a closed 3-manifold $\mathcal{M}$ with n tetrahedra, $\mathbf{v}$ vertices, $\mathrm{tw}(\Gamma(\mathcal{T})) = \mathbf{tw}$, and maximum edge valence $\Delta(\mathcal{T}) = \boldsymbol{\Delta}$, yields a triangulation $\mathcal{T}^*$ of $\mathcal{M}$ with $O((n+\mathbf{v})\log_2(\boldsymbol{\Delta})^{5.32})$ vertices and tetrahedra, $\mathrm{tw}(\Gamma(\mathcal{T}^*)) < (\mathbf{tw}+1)\cdot\log_2(\boldsymbol{\Delta})^{5.17}$, and $\Delta(\mathcal{T}^*) \leq 9$. The algorithm runs in $O(n \log\log n)$ time.*

The construction in Theorem 5 is based on an iterative application of the following result.

▶ **Theorem 6.** *There is an algorithm which, given a triangulation $\mathcal{T}$ of a closed 3-manifold $\mathcal{M}$ with n tetrahedra, $\mathbf{v}$ vertices, $\mathrm{tw}(\Gamma(\mathcal{T})) = \mathbf{tw}$, and $\Delta(\mathcal{T}) = \boldsymbol{\Delta}$, constructs a triangulation $\mathcal{T}^*$ of $\mathcal{M}$ with at most $(28+4\sqrt{6})n + (16+4\sqrt{6})\mathbf{v}$ tetrahedra, $(6+\sqrt{6})(n+\mathbf{v})$ vertices, $\Delta(\mathcal{T}^*) \leq \max\{\lfloor\sqrt{\boldsymbol{\Delta}}\rfloor + 4, 9\}$, and $\mathrm{tw}(\Gamma(\mathcal{T}^*)) \leq 36\cdot\mathbf{tw}$. The algorithm runs in $O(n)$ time.*

The proof of Theorem 6 is the core of this section. We prove Theorem 5 at the end of it.

Setup. Let us fix a triangulation $\mathcal{T}$ of a closed 3-manifold $\mathcal{M}$ with n tetrahedra, $|\mathcal{T}(0)| = \mathbf{v}$ vertices, treewidth $\mathrm{tw}(\Gamma(\mathcal{T})) = \mathbf{tw}$, and maximum edge valence $\Delta(\mathcal{T}) = \boldsymbol{\Delta}$. Let $S = S(\mathcal{T})$ be the *spine of* $\mathcal{M}$ *dual to* $\mathcal{T}$. It is a 2-dimensional polyhedral cell complex whose 1-skeleton $S^{(1)}$ is the dual graph $\Gamma(\mathcal{T})$, and for each edge $e \in \mathcal{T}(1)$, S contains a polygonal 2-face F_e glued along the cycle in $\Gamma(\mathcal{T})$ formed by the vertices corresponding to the tetrahedra that contain e. Locally, at each tetrahedron of $\mathcal{T}$, the spine S looks like the configuration shown in Figure 7.

▶ Remark. Spines, and in particular *special spines*, offer a perspective on 3-manifolds that is dual to that provided by triangulations. We refer to [32, Chapter 1] and [37] for details.

Overall strategy. Note that, given the spine S, we can recover $\mathcal{M}$ by first thickening S and then filling in the resulting 2-sphere boundary components with 3-balls (which may be viewed as centered at the vertices of $\mathcal{T}$). In what follows, we triangulate S and fill in these missing balls combinatorially. Throughout, the term 2-*face* refers to the faces of the spine S, whereas *triangle* refers to the faces of the triangulation. Our strategy is similar to that of [13, Theorem 3.36], but we employ a different – and necessarily more intricate – triangulation of the 2-faces of S in order to control the treewidth of the resulting triangulation.

Triangulating the faces of S. Consider a 2-face $F := F_e$ of the spine S transverse to an edge $e \in \mathcal{T}(1)$. This face F is a k-gon, where $k = \mathrm{val}(e)$, whose vertices are the centers of the tetrahedra of $\mathcal{T}$ containing e. We triangulate F as follows (see Figure 8). If $k \leq 9$, we add a vertex w at the center of F and connect it to all boundary vertices, as in Figure 8 (left). If $k > 9$, we proceed as follows. Fix an arbitrary orientation of the boundary of F, and choose positive integers $m := m_F$ and $d := d_F$ such that $k \leq m(d-1)$. Label the boundary vertices

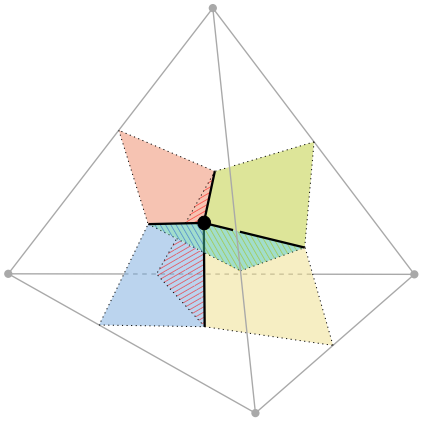


Figure 7 The intersection of the spine S with a tetrahedron of $\mathcal{T}$. The vertices of S bijectively correspond to the centers of tetrahedra of $\mathcal{T}$, and each 2-face of S is transverse to an edge of $\mathcal{T}$.

$u_0, \dots, u_{k-1}$ in accordance with this orientation. Introduce a ring of m interior vertices $v_0, \dots, v_{m-1}$, and connect each v_i to v_{i-1} and v_{i+1} (indices $\mod m$). For each $0 \le i < m-1$, connect v_i to the boundary vertices $u_{i(d-1)}, \dots, u_{(i+1)(d-1)}$. The final interior vertex v_{m-1} is connected to $u_{(m-1)(d-1)}, \dots, u_{k-1}, u_0$. The inequalities $(m-1)(d-1) < k \le m(d-1)$ ensure that the vertices u_i and v_ℓ, together with these edges, form a triangulated annulus. Finally, we place a central vertex w inside F and connect it to all v_i, see Figure 8 (right).

The next proposition follows directly from this construction; k, m, and d are as above.

▶ **Proposition 7.** *The triangulation of the 2-face F described above consists of $k + 2m$ triangles. Each boundary vertex u_i is incident to 3 or 4 edges, each interior vertex v_j is incident to at most $d+3$ edges, and the central vertex w is incident to m edges. In particular, the maximum degree of a vertex in the 1-skeleton of F is $\max\{m, d+3\}$.*

Building the triangulation $\mathcal{T}^*$ in Theorem 6. Let $\mathcal{T}_S$ denote the triangulation of the spine S obtained by triangulating each 2-face F of S as described above. We construct the triangulation $\mathcal{T}^*$ by first taking $\mathcal{T}_S$, considered as embedded in the geometric realization $\|\mathcal{T}\|$ of $\mathcal{T}$, and then adding the vertices $\mathcal{T}(0)$. Each vertex $a \in \mathcal{T}(0)$ lies in the interior of a connected component of $\|\mathcal{T}\| \setminus S$, which is an open 3-ball denoted by B_a. Its boundary $\partial B_a := \overline{B_a} \setminus B_a$ is a subpolyhedron of S comprised of certain 2-faces of S. Let $\mathcal{T}_{\partial B_a}$ denote the subtriangulation of $\mathcal{T}_S$ formed by the union of the triangulations of these 2-faces of S. To complete the construction, for each $a \in \mathcal{T}(0)$ we cone over $\mathcal{T}_{\partial B_a}$ from the vertex a. We denote the resulting triangulation by $\mathcal{T}^*$. By construction, $\mathcal{T}^*$ triangulates $\mathcal{M}$.

▶ Remark. In the above construction, for each 2-face F we fix integers $m = m_F$ and $d = d_F$ that depend solely on the number of boundary vertices u_i of F in S, or equivalently, on the valence of the edge of $\mathcal{T}$ transverse to F. When the context requires, we write d_F and m_F.

▶ **Definition 8** (type of a vertex of $\mathcal{T}^*$)**.** *The vertices of $\mathcal{T}^*$ fall into the following three* types. **1.** *vertices $a \in \mathcal{T}(0)$,* **2.** *vertices u_i lying on the boundary of exactly six 2-faces of S, and* **3.** *vertices v_ℓ and w lying in the interior of exactly one 2-face of S.*

We proceed with investigating the maximum edge valence and the treewidth of $\mathcal{T}^*$.

▶ **Lemma 9.** *The triangulation $\mathcal{T}^*$ has edge valences at most $\max_{F \in S} \max\{d_F + 3, m_F, 9\}$.*

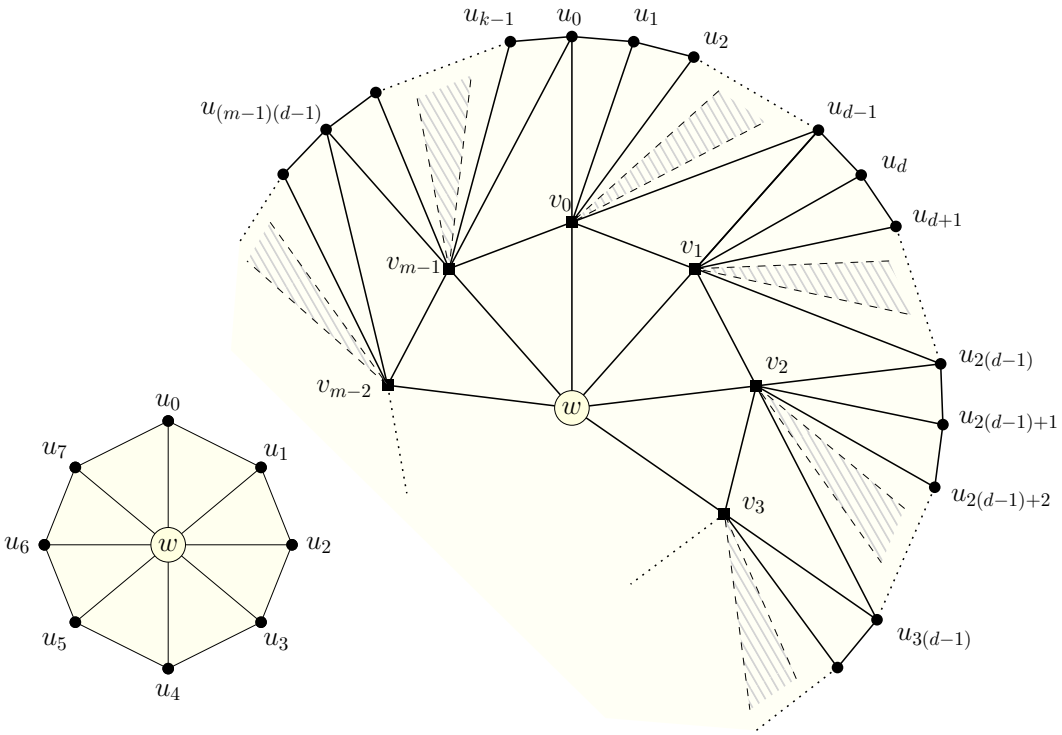


Figure 8 Triangulating a 2-face F (a k-gon) of the spine S of $\mathcal{T}$ if $k \leq 9$ (left) and if $k > 9$ (right).

Proof. We consider, case by case, the different types of edges in $\mathcal{T}^*$. Let a denote a vertex of $\mathcal{T}(0)$, and let u_i, v_ℓ, and w denote vertices in a 2-face F, see Figure 8. (We keep this labeling convention throughout this section.) The next valence-bounds follow from Proposition 7.

Type $\{a, w\}$. The valence of the edge of this type is equal to the degree of w in the triangulation of the 2-face, i.e., $\mathrm{val}(\{a, w\}) = m$.

Type $\{a, v_\ell\}$. The valence of the edge of this type is equal to the degree of v_ℓ in the triangulation of the 2-face, i.e., $d + 3$.

Type $\{a, u_i\}$. is incident to at most 3 tetrahedra per 2-face containing u_i, and there are at most three 2-faces containing u_i and bounding the ball containing a, so $\mathrm{val}(\{a, u_i\}) \leq 9$.

Type $\{w, v_\ell\}$ or $\{v_\ell, u_i\}$. are incident to at most 4 tetrahedra, two on either side of F,

Type $\{u_i, u_{i+1}\}$. is incident to one tetrahedra $\{a, v_\ell, u_i, u_{i+1}\}$ per 2-face that contains it, and is contained in at most three 2-faces, therefore $\mathrm{val}(\{u_i, u_{i+1}\}) \leq 3$.

Hence, for the maximum edge valence of $\mathcal{T}^*$ we have $\Delta(\mathcal{T}^*) \leq \max\{d_F + 3, m_F, 9\}$. ◀

▶ **Lemma 10.** *The dual graph of the triangulation $\mathcal{T}^*$ has treewidth* $\mathrm{tw}(\Gamma(\mathcal{T}^*)) < 36 \cdot (\mathbf{tw} + 1)$.

Proof. Consider a tree decomposition $\mathcal{X} = (T, \mathcal{B})$ of the graph $\Gamma(\mathcal{T})$ with optimal width $\mathbf{tw}$.

By construction, every tetrahedron of $\mathcal{T}^*$ contains a vertex of $\mathcal{T}(0)$ and at least one triangle from the triangulation of a 2-face F of S. Following the notation of Figure 8, these triangles are of the form $\{v_\ell, u_{i-1}, u_i\}$, $\{v_{\ell-1}, u_{\ell(d-1)}, v_{\ell+1}\}$, or $\{w, v_{\ell-1}, v_\ell\}$. For any 2-face of S, the boundary vertices $u_0, \ldots, u_{k-1}$ correspond to nodes of the graph $\Gamma(\mathcal{T})$, and consequently each of them appears in some bag of this tree decomposition $\mathcal{X}$.

We now construct a tree decomposition $\mathcal{X}' = (T, \mathcal{B}')$ for the dual graph of $\mathcal{T}^*$, with the same underlying tree T as for $\mathcal{X}$ and $\mathcal{B}' = \{B'_\tau : \tau \in V(T)\}$. We initialize B'_τ as $B'_\tau = \emptyset$ for all $\tau \in V(T)$, and consider all types of tetrahedra σ in $\mathcal{T}^*$:

1. If σ has a triangular face $\{v_\ell, u_{i-1}, u_i\}$, then add σ to all bags B'_τ such that $u_{i-1} \in B_\tau$.
2. If σ has a face $\{v_{\ell-1}, u_{\ell(d-1)}, v_{\ell+1}\}$, then add σ to all bags B'_τ such that $u_{\ell(d-1)} \in B_\tau$. Similarly, for $\sigma = \{a, v_{m-1}, u_0, v_0\}$, add σ to all bags B'_τ such that $u_0 \in B_\tau$.
3. If σ has a face $\{w, v_{\ell-1}, v_\ell\}$, then add σ to all bags B'_τ such that B_τ contains any u_i with $\ell(d-1) \le i < \max\{(\ell+1)(d-1), k-1\}$.

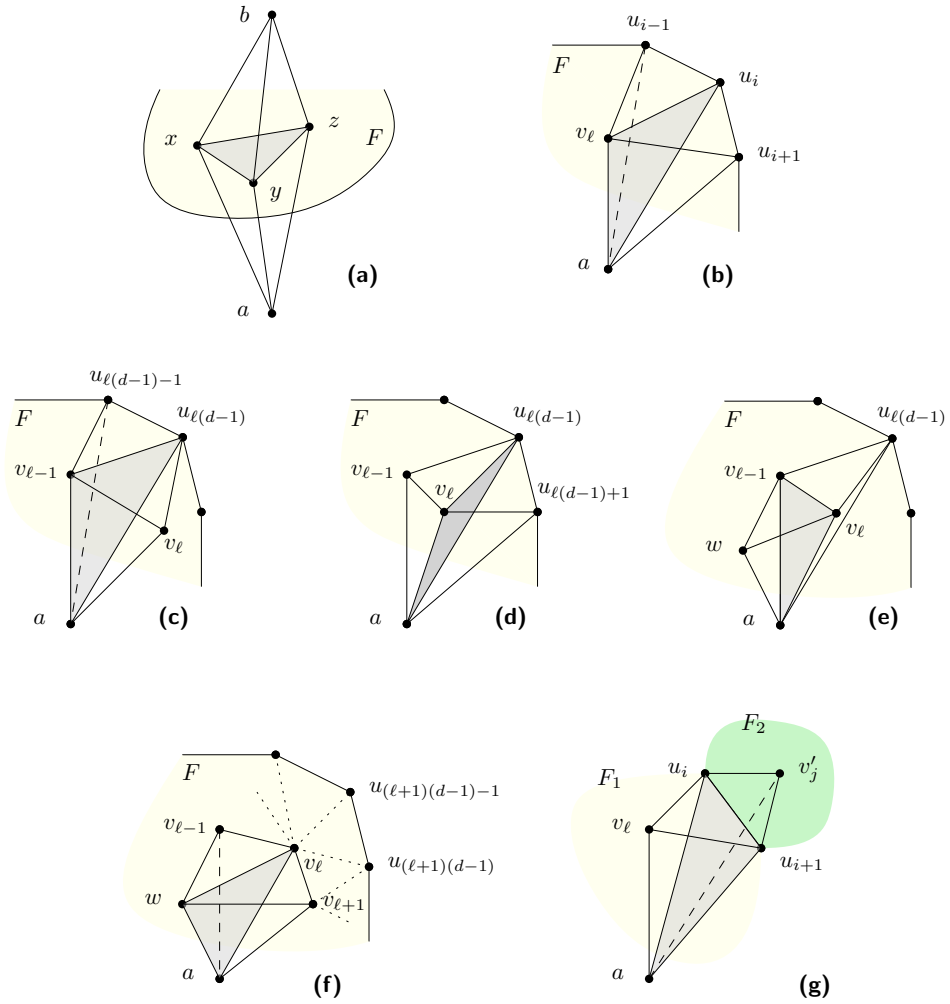


Figure 9 Illustrations of 7 possible triangles shared by tetrahedra in $\mathcal{T}^*$ from the proof of Claim 11.

▷ Claim 11. $\mathcal{X}' = (T, \mathcal{B}')$ is a valid tree decomposition of the dual graph $\Gamma(\mathcal{T}^*)$ of $\mathcal{T}^*$.

Proof. We verify the three defining properties of a tree decomposition (Section 2.1).

Vertex coverage. By construction, any tetrahedron of $\mathcal{T}^*$ contains a triangle from a 2-face F of the spine S, and in consequence appears in some bag of $\mathcal{X}' = (T, \mathcal{B}')$.

Edge coverage. Showing that every edge of $\mathcal{T}^*$ is contained in some bag of $\mathcal{X}'$ can be carried out by a tedious case analysis. In the triangulation $\mathcal{T}^*$, two tetrahedra may be adjacent

along seven types of shared triangles, classified by the types of the triangle's vertices; all possible configurations are shown in Figure 9. See the full version [18] for all the details.

Subtree property. By **1** and **2**, tetrahedra of type $\{a, v_\ell, u_i, u_{i+1}\}$, $\{a, v_{\ell-1}, u_{\ell(d-1)}, v_\ell\}$ and $\{a, v_{m-1}, u_0, v_0\}$ appear exactly in the bags B'_τ such that B_τ contain a certain fixed tetrahedron (respectively u_i, $u_{\ell(d-1)}$, and u_0); the bags containing them are consequently connected in σ. By **3**, a tetrahedron σ of type $\{a, w, v_{\ell-1}, v_\ell\}$ is inserted in all bags B'_τ such that B_τ contains any u_i with $\ell(d-1) \leq i < (\ell+1)(d-1)$. Call T_i the subtree of T spanned by the nodes τ such that $u_i \in B_\tau$ in $\mathcal{X}$. The subtree of T spanned by the nodes τ in $\mathcal{X}'$ such that $\sigma \in B_\tau$ is, in consequence, $\cup_{\ell(d-1)\leq i<(\ell+1)(d-1)} T_i$. Because $\mathcal{X}$ is a tree decomposition, each T_i is connected. Additionally, for any i, u_i and u_{i+1} are adjacent, and must consequently appear in a common bag of $\mathcal{X}$. Hence $T_i \cap T_{i+1} \neq \emptyset$. In consequence, $\cup_{\ell(d-1)\leq i<(\ell+1)(d-1)} T_i$ is a connected subtree of σ. ◁

▷ Claim 12. $\mathcal{X}' = (T, \mathcal{B}')$ has width less than $36 \cdot (\mathbf{tw} + 1)$.

Proof. Let B_τ be a bag of $\mathcal{X}$ of size r. Every occurrence of a tetrahedron u_i in B_τ will induce the insertion of at most 6 tetrahedra in B'_τ, per 2-face incident to u_i, namely tetrahedra $(\star, v_\ell, u_i, u_{i+1}\}$, tetrahedra $(\star, v_\ell, u_i, v_{\ell+1}\}$ if $i = \ell(d-1)$ or $i = 0$, and tetrahedra $(\star, w, v_{\ell-1}, v_\ell\}$ with $\ell(d-1) \leq i \leq \max\{(\ell+1)(d-1)-1, k-1\}$; where $\star$ indicates at most two possible vertices a and b of $\mathcal{T}(0)$, on either side of the 2-face. Adding the fact that at most six 2-faces are incident to a given tetrahedron u_i, the size of a bag B'_τ is at most 36 times the size of B_τ. In consequence, the width of $\mathcal{X}'$ is at most $36 \cdot (\mathbf{tw} + 1) - 1$. This concludes the proofs of Claim 12 and Lemma 10. ◁

From now on, for the 2-face F_e transverse to the edge $e \in \mathcal{T}(1)$ with valence $k = \mathrm{val}(e)$, we set $m := \lfloor\sqrt{\mathrm{val}(e)}\rfloor + 4$ and $d := \lfloor\sqrt{\mathrm{val}(e)}\rfloor + 1$, which satisfy the condition $k \leq m(d-1)$ whenever $\mathrm{val}(e) > 6$. The next two lemmas provide bounds on the number of tetrahedra (Lemma 13) and the number of vertices (Lemma 14) in $\mathcal{T}^*$. These rely on standard counting arguments for triangulations; see the full version [18].

▶ **Lemma 13.** *The triangulation $\mathcal{T}^*$ has at most $(28 + 4\sqrt{6})n + (16 + 4\sqrt{6})\mathbf{v}$ tetrahedra.*

Proof. Each 2-face, transversal to an edge e of valence $\mathrm{val}(e)$, contributes to $2(k + 2m) = 2\,\mathrm{val}(e) + 4(\lfloor\sqrt{\mathrm{val}(e)}\rfloor + 4)$ tetrahedra ($k + 2m$ triangles in the transversal 2-face F, holding a tetrahedron on either side of F). In consequence,

$$\sum_{e\in\mathcal{T}(1)} (2\,\mathrm{val}(e) + 4(\lfloor\sqrt{\mathrm{val}(e)}\rfloor + 4)) \underbrace{\leq}_{\text{[18, Lem. 24, Cor. 25]}} 12n + 16(n+\mathbf{v}) + 4\sum_e \sqrt{\mathrm{val}(e)} \underbrace{\leq}_{\text{[18, Lem. 26]}} 12n + 16(n+\mathbf{v}) + 4\sqrt{6}(n+\mathbf{v}).$$

◀

▶ **Lemma 14.** *The triangulation $\mathcal{T}^*$ has at most $(6 + \sqrt{6})(n + \mathbf{v})$ vertices.*

Proof. By construction, the vertex set $\mathcal{T}^*(0)$ consist of the vertices $\mathcal{T}(0)$, the vertices u_i (lying on the boundaries of the 2-faces), and, for each $e \in \mathcal{T}(1)$, the $\lfloor\sqrt{\mathrm{val}(e)}\rfloor + 5$ interior vertices on the triangulated 2-face F_e. The vertices u_i are in bijection with the tetrahedra of $\mathcal{T}$, hence there are exactly n of them. The total number of interior vertices over all 2-faces is

$$\sum_{e\in\mathcal{T}(1)} \left(\lfloor\sqrt{\mathrm{val}(e)}\rfloor + 5\right) \underbrace{\leq}_{\text{[18, Lemmas 24, 26]}} (\sqrt{6}+5)(n+\mathbf{v}).$$

With $|\mathcal{T}(0)| = \mathbf{v}$ and the n vertices u_i, this gives at most $(6 + \sqrt{6})(n + \mathbf{v})$ vertices in $\mathcal{T}^*$. ◀

We are now ready to prove the main theorems of this section.

Proof of Theorem 6. If the maximum edge valence of $\mathcal{T}$ is at most 9, the algorithm simply returns $\mathcal{T}^* := \mathcal{T}$. Otherwise, we apply the construction described above; the bounds on the number of tetrahedra, the number of vertices, the treewidth, and the edge valence follow from Lemmas 13, 14, 10, and 9, respectively. Note that the construction runs in time linear in the size of $\mathcal{T}^*$, which is $O(n)$. ◀

Finally, the proof of Theorem 5 relies on standard calculations involving geometric series and iterated radicals, as detailed in the full version [18].

Proof of Theorem 5. If the maximum edge valence of $\mathcal{T}$ satisfies $\Delta(\mathcal{T}) \leq 9$, we are done. Otherwise, we construct a sequence of triangulations $\mathcal{T} = \mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_p$ by iterating the construction of Theorem 6. The triangulation $\mathcal{T}_r$ has $O((30+4\sqrt{6})^r(n+\mathbf{v}))$ tetrahedra and vertices [18, Lemma 23], treewidth $\mathrm{tw}(\Gamma(\mathcal{T}_r)) < 36^r \cdot (\mathbf{tw}+1)$ (Lemma 10), and maximum edge valence $\Delta(\mathcal{T}_r) \leq \max\{a_r, 9\}$ (Lemma 9, with $m = \lfloor\sqrt{\mathrm{val}(e)}\rfloor + 4$ and $d = \lfloor\sqrt{\mathrm{val}(e)}\rfloor + 1$), where $(a_r)_{r\in\mathbb{N}}$ is the sequence of iterated radicals of [18, Lemma 21], with $a_0 = \mathbf{\Delta}$, $c = 4$, and $L < 6.32$. By [18, Lemma 22], choosing $p = \log_2 \log_2(\mathbf{\Delta})$ ensures that the edge valence of $\mathcal{T}_p$ is at most 9. We therefore set $\mathcal{T}^* := \mathcal{T}_p$. Since $\log_2(30+4\sqrt{6}) < 5.32$ and $\log_2(36) < 5.17$, the triangulation $\mathcal{T}^*$ has $O((n+\mathbf{v})\log_2(\mathbf{\Delta})^{5.32})$ tetrahedra and vertices, treewidth at most $(\mathbf{tw}+1)\cdot\log_2(\mathbf{\Delta})^{5.17}$, and edge valence at most 9. The algorithm performs $O(\log\log(n))$ iterations of the algorithm of Theorem 6, yielding the stated complexity bound. ◀

5 Efficient evaluation of Kuperberg's tensor networks

In this section we present a fixed-parameter tractable (FPT) scheme for computing Kuperberg's quantum invariants $\sharp(\vec{\mathcal{D}}, \mathcal{H})$ of closed, oriented[10] 3-manifolds [23]. This quantity depends on an oriented Heegaard diagram[11] $\vec{\mathcal{D}} = (\vec{\mathcal{S}}, \vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}})$ and a finite-dimensional involutory Hopf algebra $\mathcal{H}$ over a field $\mathbb{F}$, which is considered fixed. By construction, $\sharp(\vec{\mathcal{D}}, \mathcal{H}) \in \mathbb{F}$.

▶ Remark. We emphasize that understanding the results in this section does **not** require knowledge of Hopf algebras or tensors, since the following proofs are purely graph theoretical. Nevertheless, we provide some basic definitions and examples in the full version [18]. We refer to Kuperberg's paper [23] (cf. its open-access version [22]) for further details.

Even if $\sharp(\vec{\mathcal{D}}, \mathcal{H})$ is originally derived from an oriented Heegaard diagram, it can also be computed from an oriented triangulation. This is briefly hinted at in [23, Section 4], however, with the machinery developed in Section 3 we can make it fully explicit: Any triangulation $\mathcal{T}$ of a 3-manifold $\mathcal{M}$ gives rise to a Heegaard diagram $\mathcal{D} = (\mathcal{S}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ of $\mathcal{M}$, moreover, in a width-preserving way (Theorem 1). Picking an orientation for $\mathcal{T}$ induces an orientation also for $\mathcal{S}$. Then choosing orientations for the α- and β-curves arbitrarily yields an oriented diagram $\vec{\mathcal{D}}$. Next, from $\vec{\mathcal{D}}$ we build a *tensor network* $\mathcal{K}$ following Kuperberg's construction [23, Section 5], which we finally *evaluate* to obtain $\sharp(\vec{\mathcal{D}}, \mathcal{H})$. From now on, we will use the notation $\sharp(\vec{\mathcal{T}}, \mathcal{H}) := \sharp(\vec{\mathcal{D}}, \mathcal{H})$ to indicate when we regard the invariant as computed from a triangulation, where $\vec{\mathcal{D}}$ denotes the oriented Heegaard diagram induced by $\vec{\mathcal{T}}$.

[10] In this section we assume $\mathcal{M}$ to be an *oriented*, not merely orientable, 3-manifold.

[11] An *oriented Heegaard diagram* is a Heegaard diagram $\mathcal{D} = (\mathcal{S}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ together with an orientation for the surface $\mathcal{S}$ and for each of the α- and β-curves. The quantity $\sharp(\vec{\mathcal{D}}, \mathcal{H})$ does not depend on the choice of orientation for the α- and β-curves, however, it generally does depend on the chosen orientation for $\mathcal{S}$.

Building on Theorem 1 and bounding the treewidth of the tensor network $\mathcal{K}$ (Lemma 18), together with work of O'Gorman [34] (cf. Markov–Shi [31]) we prove the following result:

▶ **Theorem 15.** *Let $\mathcal{H}$ be a fixed finite-dimensional involutory Hopf algebra over a field $\mathbb{F}$. Let $\vec{\mathcal{T}}$ be an oriented triangulation of a closed oriented 3-manifold with n tetrahedra and treewidth $\mathrm{tw}(\vec{\mathcal{T}}) := \mathrm{tw}(\Gamma(\vec{\mathcal{T}})) = t$. The invariant $\sharp(\vec{\mathcal{T}}, \mathcal{H})$ can be computed in time $O(2^{O(t)}n)$.*

Before discussing the proof of Theorem 15 in detail, we provide an overview on Figure 10.

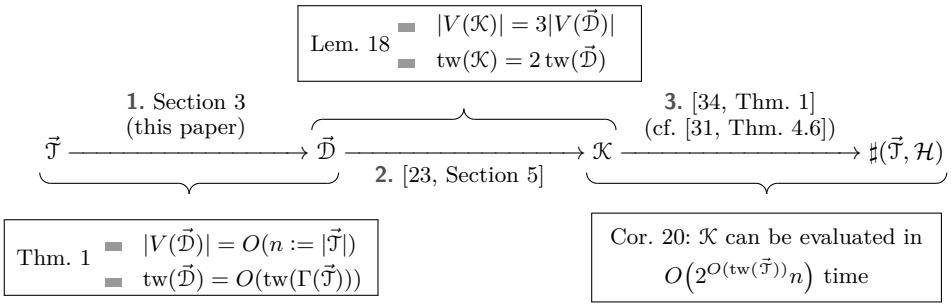


Figure 10 The algorithmic pipeline of computing $\sharp(\vec{\mathcal{T}}, \mathcal{H})$ from an oriented triangulation $\vec{\mathcal{T}}$. The references on the arrows provide the theoretical backbone, and the framed results indicate our contributions. The running time of the computation is dominated by that of step **3.** $\mathcal{K} \longrightarrow \sharp(\vec{\mathcal{T}}, \mathcal{H})$.

▶ Remark 16. Theorem 15 asserts that computing $\sharp(\vec{\mathcal{T}}, \mathcal{H})$ is FPT in the treewidth of $\vec{\mathcal{T}}$. In particular, for bounded-treewidth triangulations it can be computed in linear time. However, since tensor network contraction – the last part of the computation – is generally $\sharp\mathbf{P}$-hard [10], we expect this to hold for computing $\sharp(\vec{\mathcal{T}}, \mathcal{H})$ for general triangulations as well.

As motivation, let us recall the formula for the invariant $\sharp(\vec{\mathcal{D}}, \mathcal{H})$ from [23, Section 5]:

$$\sharp(\vec{\mathcal{D}}, \mathcal{H}) := \mathcal{Z}(\vec{\mathcal{D}}, \mathcal{H}) \cdot \dim_{\mathbb{F}}(\mathcal{H})^{g(\mathcal{S})-|\boldsymbol{\alpha}|-|\boldsymbol{\beta}|}. \tag{2}$$

Here $g(\mathcal{S})$ is the genus of the surface $\mathcal{S}$, and $|\boldsymbol{\alpha}|$ and $|\boldsymbol{\beta}|$ denote the number of α- and β-curves of $\vec{\mathcal{D}}$, respectively. The "complicated" term $\mathcal{Z}(\vec{\mathcal{D}}, \mathcal{H}) \in \mathbb{F}$ is defined as the *evaluation* $\mathcal{Z}(\mathcal{K})$ of a specific *tensor network* $\mathcal{K}$ associated to $\vec{\mathcal{D}}$, built from (several copies of) the *structure tensors* of the Hopf algebra $\mathcal{H}$. We now very briefly and selectively review these notions.

Hopf algebras and their structure tensors. A *Hopf algebra* $\mathcal{H}$ is a vector space $\boldsymbol{H}$ over a field $\mathbb{F}$ together with three $\mathbb{F}$-linear maps $\mathsf{M}\colon \boldsymbol{H} \otimes \boldsymbol{H} \to \boldsymbol{H}$ (*multiplication*), $\Delta\colon \boldsymbol{H} \to \boldsymbol{H} \otimes \boldsymbol{H}$ (*comultiplication*) and $\mathsf{S}\colon \boldsymbol{H} \to \boldsymbol{H}$ (*antipode*) satisfying certain axioms (cf. [23, Section 3]). We will assume that $\dim_{\mathbb{F}}(\boldsymbol{H}) < \infty$ and $\mathcal{H}$ is *involutory*, i.e., $\mathsf{S} \circ \mathsf{S} = \mathrm{id}_{\boldsymbol{H}}$. The maps M, Δ and S can be regarded as tensors of type $(1,2)$, $(2,1)$ and $(1,1)$, respectively, see Figure 11.

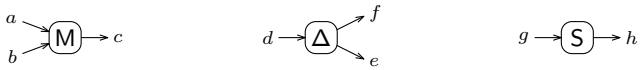


Figure 11 The multiplication, comultiplication and antipode tensors represented as *coupons*.

▶ **Example 17.** The *group algebra* $\mathcal{G} = \mathbb{F}G$ of a finite group G is naturally a Hopf algebra: the multiplication is defined as $\mathsf{M}(g \otimes h) = gh$, the comultiplication as $\Delta(g) = g \otimes g$, and the antipode as $\mathsf{S}(g) = g^{-1}$. In [23, Sections 1 and 6] it is shown that $\sharp(\vec{\mathcal{D}}, \mathcal{G})$ equals the number of homomorphisms $\pi_1(\mathcal{M}) \to G$ from the fundamental group of the 3-manifold $\mathcal{M}$ described by $\vec{\mathcal{D}}$ to the group G. (In this case $\sharp(\vec{\mathcal{D}}, \mathcal{G})$ is independent of the chosen orientation for $\mathcal{D}$.)

5.1 Kuperberg's construction and the proof of Theorem 15

Following [23, Section 5], first we explain how the tensor network $\mathcal{K}$ – whose evaluation $\mathcal{Z}(\mathcal{K}) =: \mathcal{Z}(\vec{\mathcal{D}}, \mathcal{H})$ appears in (2) – is constructed from an oriented Heegaard diagram $\vec{\mathcal{D}}$. Then we prove Lemma 18, which – together with Theorem 19 – implies Theorem 15.

The construction of $\mathcal{K}$. As already mentioned, the building blocks of $\mathcal{K}$ are the structure tensors M, Δ and S of the Hopf algebra $\mathcal{H}$ (Figure 11). The shorthands in Figure 12 – which are justified by the fact that both abbreviated tensor networks, called the *tracial product* and *tracial coproduct* in [23], are cyclically symmetric[12] – will be useful:



Figure 12 Diagrammatic abbreviations of the *tracial product* and *tracial coproduct* tensors.

Let $\vec{\mathcal{D}} = (\vec{\mathcal{S}}, \vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}})$ be an oriented Heegaard diagram. For every oriented α-curve $\vec{\alpha}_i \in \vec{\boldsymbol{\alpha}}$, we take a copy of the tracial product tensor  , where the abstract indices $c_1, \ldots, c_p$ correspond to the crossings on $\vec{\alpha}_i$ (with the β-curves of $\vec{\mathcal{D}}$) in the order they are encountered when traversing $\vec{\alpha}_i$ following its orientation. We assign a copy  of the tracial coproduct tensor to each oriented β-curve $\vec{\beta}_j \in \vec{\boldsymbol{\beta}}$ analogously. Note that each crossing c between an α- and a β-curve of $\vec{\mathcal{D}}$ appears exactly twice as an abstract index of the above tensors: once at a product and once at a coproduct. The construction of $\mathcal{K}$ is concluded by contracting these indices (i.e., joining the respective arrows with *wires*) with a caveat: if the tangent vectors of the β- and α-curves intersecting at c form (in that order) a negatively oriented basis of the tangent space $T\vec{\mathcal{S}}$, then the antipode tensor is interposed before contracting (Figure 13).

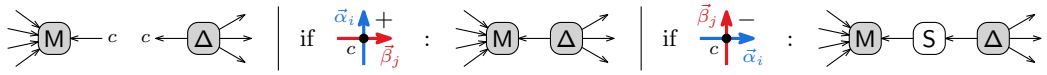


Figure 13 Instruction manual for assembling the tensor network $\mathcal{K}$ from its building blocks.

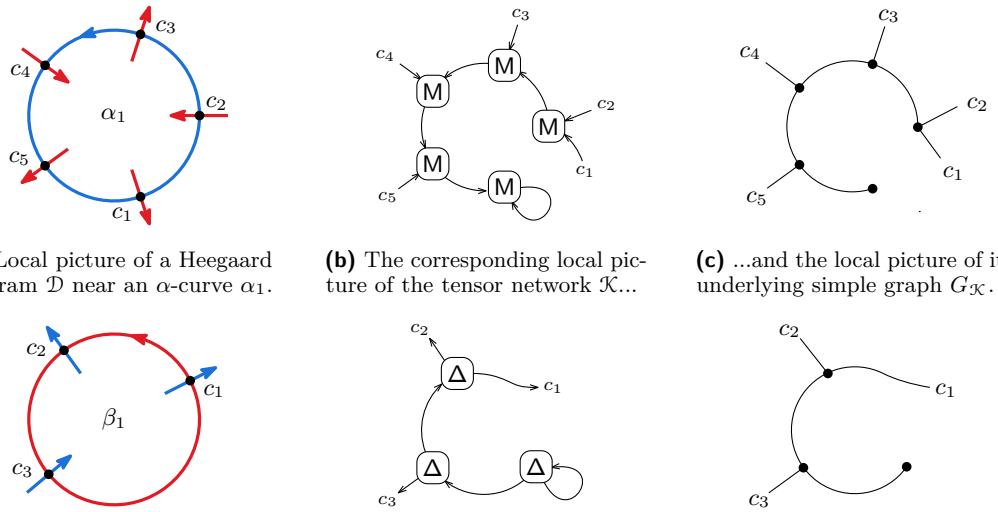
A fully contracted tensor network $\mathcal{N}$ (such as $\mathcal{K}$) can be seen as a graph $(V(\mathcal{N}), E(\mathcal{N}))$ in the obvious way: $V(\mathcal{N})$ corresponds to the coupons of $\mathcal{N}$, whereas $E(\mathcal{N})$ to its wires. (We emphasize that $\mathcal{N}$ is considered here without any diagrammatic abbreviations.)

▶ **Lemma 18.** *Let $\vec{\mathcal{D}}$ be an oriented Heegaard diagram and $\mathcal{K}$ be the Kuperberg tensor network induced by $\vec{\mathcal{D}}$. Then, the number of vertices and the treewidth of $\vec{\mathcal{D}}$ and $\mathcal{K}$ satisfy*

$$|V(\mathcal{K})| \leq 3|V(\vec{\mathcal{D}})| \qquad \text{(3a)} \qquad \textit{and} \qquad \operatorname{tw}(\mathcal{K}) \leq 2\operatorname{tw}(\vec{\mathcal{D}}). \qquad \text{(3b)}$$

Proof. Both inequalities (3a) and (3b) follow directly from the construction of $\mathcal{K}$. In fact, the local configurations shown in Figure 14 already reveal the close structural relationship between $\vec{\mathcal{D}}$ and $\mathcal{K}$. We refer to the full version [18] for the complete proof. ◀

[12] This is analogous to the fact that $\operatorname{Tr}(A_1 A_2 \cdots A_n) = \operatorname{Tr}(A_n A_1 \cdots A_{n-1})$ for the trace of matrices.



(a) Local picture of a Heegaard diagram $\mathcal{D}$ near an α-curve α_1.

(b) The corresponding local picture of the tensor network $\mathcal{K}$...

(c) ...and the local picture of its underlying simple graph $G_{\mathcal{K}}$.

(d) Local picture of a Heegaard diagram $\mathcal{D}$ near an β-curve β_1.

(e) The corresponding local picture of the tensor network $\mathcal{K}$...

(f) ...and the local picture of its underlying simple graph $G_{\mathcal{K}}$.

Figure 14 The local structural correspondence between a Heegaard diagram $\mathcal{D}$ and the induced Kuperberg tensor network $\mathcal{K}$.

Proof of Theorem 15. With Theorem 1 and Lemma 18 at hand, Theorem 15 is a consequence of the following general result about the computational cost of evaluating tensor networks.

▶ **Theorem 19** ([34, Theorem 1], cf. [31, Theorem 4.6]). *Any tensor network $\mathcal{N}$ with n tensors can be evaluated in $O\big(2^{O(\mathrm{vc}(\mathcal{N}))}n\big)$ time, where $\mathrm{vc}(\mathcal{N})$ denotes the* vertex congestion *of $\mathcal{N}$.*

Vertex congestion is a graph parameter similar to treewidth, discussed in [1, p. 135]. For a graph G with maximum degree Δ, we have $\mathrm{vc}(G) \leq (3/2)\cdot\Delta\cdot(\mathrm{tw}(G)+1)$, see [1, Theorem 1 and Remark 3]. Since Kuperberg's tensor networks are 3-regular graphs, we have

▶ **Corollary 20.** *Any Kuperberg tensor network $\mathcal{K}$ can be evaluated in time $O\big(2^{O(\mathrm{tw}(\mathcal{K}))}|\mathcal{K}|\big)$, where $\mathrm{tw}(\mathcal{K})$ denotes the treewidth and $|\mathcal{K}|$ the number of tensors of $\mathcal{K}$.*

Now, if $\mathcal{K}$ is induced by an oriented triangulation $\vec{\mathcal{T}}$ with n tetrahedra, then $|\mathcal{K}| \leq 3|V(\vec{\mathcal{D}})|$ and $\mathrm{tw}(\mathcal{K}) \leq 2\,\mathrm{tw}(\vec{\mathcal{D}})$ by Lemma 18, and $|V(\vec{\mathcal{D}})| \leq 6$ and $\mathrm{tw}(\vec{\mathcal{D}}) \leq 12\,\mathrm{tw}(\vec{\mathcal{T}})+11$ by Theorem 1. Combining these inequalities with Corollary 20, the Theorem 15 follows. ◀

References

1 D. Bienstock. On embedding graphs in trees. *J. Comb. Theory, Ser. B*, 49(1):103–136, 1990. doi:10.1016/0095-8956(90)90066-9.

2 N. Brady, J. McCammond, and J. Meier. Bounding edge degrees in triangulated 3-manifolds. *Proc. Amer. Math. Soc.*, 132(1):291–298, 2004. doi:10.1090/S0002-9939-03-06981-8.

3 B. A. Burton. The HOMFLY-PT polynomial is fixed-parameter tractable. In *34th Int. Symp. Comput. Geom. (SoCG 2018)*, volume 99 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 18:1–18:14. Schloss Dagstuhl–Leibniz-Zent. Inf., 2018. doi:10.4230/LIPIcs.SoCG.2018.18.

4 B. A. Burton, R. Budney, W. Pettersson, et al. Regina: Software for low-dimensional topology, 1999–2025. Version 7.4.1. URL: https://regina-normal.github.io.

5 B. A. Burton and R. G. Downey. Courcelle's theorem for triangulations. *J. Comb. Theory, Ser. A*, 146:264–294, 2017. doi:10.1016/j.jcta.2016.10.001.

6 B. A. Burton, T. Lewiner, J. Paixão, and J. Spreer. Parameterized complexity of discrete Morse theory. *ACM Trans. Math. Softw.*, 42(1):6:1–6:24, 2016. doi:10.1145/2738034.

7 B. A. Burton, C. Maria, and J. Spreer. Algorithms and complexity for Turaev–Viro invariants. *J. Appl. Comput. Topol.*, 2(1–2):33–53, 2018. doi:10.1007/s41468-018-0016-2.

8 B. A. Burton and W. Pettersson. Fixed parameter tractable algorithms in combinatorial topology. In *Proc. 20th Int. Conf. Comput. Comb. (COCOON 2014)*, volume 8591 of *Lect. Notes Comput. Sci.*, pages 300–311. Springer, 2014. doi:10.1007/978-3-319-08783-2_26.

9 B. A. Burton and J. Spreer. The complexity of detecting taut angle structures on triangulations. In *Proc. 24th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA 2013)*, pages 168–183, 2013. doi:10.1137/1.9781611973105.13.

10 C. Damm, M. Holzer, and P. McKenzie. The complexity of tensor calculus. *Comput. Complexity*, 11(1-2):54–89, 2002. doi:10.1007/s00037-000-0170-4.

11 A. de Mesmay, J. Purcell, S. Schleimer, and E. Sedgwick. On the tree-width of knot diagrams. *J. Comput. Geom.*, 10(1):164–180, 2019. doi:10.20382/jocg.v10i1a6.

12 D. Eppstein. What is ... treewidth? *Notices Amer. Math. Soc.*, 72(2):172–175, 2025. URL: https://www.ams.org/journals/notices/202502/rnoti-p172.pdf.

13 F. Frick. *Combinatorial restrictions on cell complexes*. PhD thesis, TU Berlin, 2015. doi:10.14279/depositonce-4545.

14 A. He, J. Morgan, and E. K. Thompson. An algorithm to construct one-vertex triangulations of Heegaard splittings. *J. Comput. Geom.*, 16(1):635–693, November 2025. doi:10.20382/jocg.v16i1a18.

15 P. Heegaard. *Forstudier til en topologisk Teori for de algebraiske Fladers Sammenhæng* (in Danish). PhD thesis, Det Nordiske Forlag, 1898. URL: https://books.google.com/books?id=p9hMAAAAYAAJ. An English translation is available at https://webhomes.maths.ed.ac.uk/~v1ranick/papers/heegaardenglish.pdf.

16 P. Heegaard. Sur l'"Analysis situs". *Bull. Soc. Math. France*, 44:161–242, 1916. doi:10.24033/bsmf.968.

17 K. Huszár. On the pathwidth of hyperbolic 3-manifolds. *Comput. Geom. Topol.*, 1(1):1–19, 2022. doi:10.57717/cgt.v1i1.4.

18 K. Huszár and C. Maria. On sparse representations of 3-manifolds, 2025. arXiv:2512.05779.

19 K. Huszár and J. Spreer. 3-Manifold triangulations with small treewidth. In *35th Int. Symp. Comput. Geom. (SoCG 2019)*, volume 129 of *LIPIcs. Leibniz Int. Proc. Inf.*, pages 44:1–44:20. Schloss Dagstuhl–Leibniz-Zent. Inf., 2019. doi:10.4230/LIPIcs.SoCG.2019.44.

20 K. Huszár and J. Spreer. On the width of complicated JSJ decompositions. *Discrete Comput. Geom.*, 74(4):917–943, 2025. doi:10.1007/s00454-025-00746-1.

21 K. Huszár, J. Spreer, and U. Wagner. On the treewidth of triangulated 3-manifolds. *J. Comput. Geom.*, 10(2):70–98, 2019. doi:10.20382/jogc.v10i2a5.

22 G. Kuperberg. Involutory Hopf algebras and 3-manifold invariants, 1990. arXiv:math/9201301.

23 G. Kuperberg. Involutory Hopf algebras and 3-manifold invariants. *Internat. J. Math.*, 2(1):41–66, 1991. doi:10.1142/S0129167X91000053.

24 G. Kuperberg. Noninvolutory Hopf algebras and 3-manifold invariants. *Duke Math. J.*, 84(1):83–129, 1996. doi:10.1215/S0012-7094-96-08403-3.

25 M. Lackenby. Algorithms in 3-manifold theory. In I. Agol and D. Gabai, editors, *Surveys in 3-manifold topology and geometry*, volume 25 of *Surv. Differ. Geom.*, pages 163–213. Int. Press Boston, 2022. doi:10.4310/SDG.2020.v25.n1.a5.

26 C. Lunel and A. de Mesmay. A Structural Approach to Tree Decompositions of Knots and Spatial Graphs. In *39th Int. Symp. Comput. Geom. (SoCG 2023)*, volume 258 of *LIPIcs. Leibniz Int. Proc. Inf.*, pages 50:1–50:16. Schloss Dagstuhl–Leibniz-Zent. Inf., 2023. doi:10.4230/LIPIcs.SoCG.2023.50.

27 J. A. Makowsky. Coloured Tutte polynomials and Kauffman brackets for graphs of bounded tree width. *Discrete Appl. Math.*, 145(2):276–290, 2005. doi:10.1016/j.dam.2004.01.016.

28 J. A. Makowsky and J. P. Mariño. The parametrized complexity of knot polynomials. *J. Comput. Syst. Sci.*, 67(4):742–756, 2003. Special issue on parameterized computation and complexity. doi:10.1016/S0022-0000(03)00080-1.

29 C. Maria. Parameterized complexity of quantum knot invariants. In *37th Int. Symp. Comput. Geom. (SoCG 2021)*, volume 189 of *LIPIcs. Leibniz Int. Proc. Inf.*, pages 53:1–53:15. Schloss Dagstuhl – Leibniz-Zent. Inf., 2021. doi:10.4230/LIPIcs.SoCG.2021.53.

30 C. Maria and J. Purcell. Treewidth, crushing and hyperbolic volume. *Algebr. Geom. Topol.*, 19(5):2625–2652, 2019. doi:10.2140/agt.2019.19.2625.

31 I. L. Markov and Y. Shi. Simulating quantum computation by contracting tensor networks. *SIAM J. Comput.*, 38(3):963–981, 2008. doi:10.1137/050644756.

32 S. Matveev. *Algorithmic Topology and Classification of* 3*-Manifolds*, volume 9 of *Algorithms Comput. Math.* Springer, Berlin, 2nd edition, 2007. doi:10.1007/978-3-540-45899-9.

33 E. E. Moise. Affine structures in 3-manifolds. V. The triangulation theorem and Hauptvermutung. *Ann. Math. (2)*, 56:96–114, 1952. doi:10.2307/1969769.

34 B. O'Gorman. Parameterization of tensor network contraction. In *14th Conf. Theory Quantum Comput. Commun. Cryptogr.*, volume 135 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 10:1–10:19. Schloss Dagstuhl. Leibniz-Zent. Inform., 2019. doi:10.4230/LIPIcs.TQC.2019.10.

35 T. Radó. Über den Begriff der Riemannschen Fläche. *Acta Sci. Math.*, 2(2):101–121, 1925.

36 N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

37 J. H. Rubinstein, H. Segerman, and S. Tillmann. Traversing three-manifold triangulations and spines. *Enseign. Math.*, 65(1-2):155–206, 2019. doi:10.4171/lem/65-1/2-5.

38 M. Scharlemann. Heegaard splittings of compact 3-manifolds. In *Handbook of Geometric Topology*, pages 921–953. North-Holland, 2001. doi:10.1016/B978-044482432-5/50019-6.

39 J. Schultens. *Introduction to* 3*-Manifolds*, volume 151 of *Grad. Stud. Math.* Am. Math. Soc., Providence, RI, 2014. doi:10.1090/gsm/151.

Improved Bound for the k-Variate Elekes–Rónyai Theorem

Yaara Jahn
Hebrew University of Jerusalem, Israel

Orit E. Raz
Ben-Gurion University, Be'er-Sheva, Israel

Abstract

Let $f \in \mathbb{R}[x_1, \ldots, x_k]$, for $k \geq 2$. For any finite sets $A_1, \ldots, A_k \subset \mathbb{R}$, consider the set

$$f(A_1, \ldots, A_k) := \{f(a_1, \ldots, a_k) \mid (a_1, \cdots, a_k) \in A_1 \times \cdots \times A_k\},$$

that is, the image of $A_1 \times \cdots \times A_k$ under f. Extending a theorem of Elekes and Rónyai, which deals with the case $k = 2$, and the result of Raz, Sharir, and De Zeeuw [9], dealing with the case $k = 3$, it is proved in Raz and Shem Tov [10], that for every choice of finite $A_1, \ldots, A_k \subset \mathbb{R}$, each of size n, one has

$$|f(A_1, \ldots, A_k)| = \Omega(n^{3/2}), \tag{1}$$

unless f has some degenerate special form.

In this paper, we introduce the notion of a *rank* of a k-variate polynomial f, denoted as $\mathrm{rank}(f)$. Letting $r = \mathrm{rank}(f)$, we prove that

$$|f(A_1, \ldots, A_k)| = \Omega\left(n^{\frac{5r-4}{2r}-\varepsilon}\right), \tag{2}$$

for every $\varepsilon > 0$, where the constant of proportionality depends on ε and on $\deg(f)$. This improves the lower bound (1), for polynomials f for which $\mathrm{rank}(f) \geq 3$.

We present an application of our main result, to lower bound the number of distinct d-volumes spanned by $(d+1)$-tuples of points lying on the moment curve in $\mathbb{R}^d$.

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems

Keywords and phrases Polynomial Expansion, Elekes–Rónyai theorem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.59

1 Introduction

In many cases in combinatorial geometry, counting questions involving distances, slopes, collinearity, etc., can be reformulated as analogous counting questions involving grid points lying on certain algebraic varieties. A unified study of such problems began with a question of Elekes [3] about expansion of bivariate real polynomials $f(x, y)$. Specifically, he asked: For a bivariate polynomial $f \in \mathbb{R}[x, y]$ and given finite sets $A, B \subset \mathbb{R}$, how small can be the image set

$$f(A, B) = \{f(a, b) \mid a \in A, b \in B\}.$$

Elekes conjectured that the image of f on a $n \times n$ Cartesian product must be of cardinality superlinear in n, unless f has a very concrete *special form*. This was confirmed in 2000 by Elekes and Rónyai [4] who proved the following dichotomy: Either f has one of the forms

$$f(x, y) = h(p(x) + q(y)) \quad \text{or}$$
$$f(x, y) = h(p(x)q(y)), \tag{3}$$

© Yaara Jahn and Orit E. Raz;
licensed under Creative Commons License CC-BY 4.0

for some univariate real polynomials p, q, h, or, otherwise, for every finite $A, B \subset \mathbb{R}$, each of size n, we have

$$|f(A,B)| = \omega(n). \tag{4}$$

In case f is not one of the forms in (3), the lower bound on $|f(A,B)|$ was improved in [8] to be $\Omega(n^{4/3})$, and further improved in [13] to be $\Omega(n^{3/2})$, which is currently the best known lower bound for bivariate polynomials that are not special.

An analogue of the Elekes–Rónyai problem can be formulated for polynomials in more than two variables. The trivariate case was studied by Raz, Sharir, and De Zeeuw [9], and the general k-variate case was established by Raz and Shem Tov [10]. They obtain the following result.

▶ **Theorem 1** ([9, 10]). *Let $k \geq 3$ and let $f \in \mathbb{R}[x_1, \ldots, x_k]$. Then one of the following holds:*

(i) *For every $A_1, \ldots, A_k \in \mathbb{R}$ each of size n one has*

$$|f(A_1, \ldots, A_k)| = \Omega\left(n^{3/2}\right)$$

(ii) *f is of one of the forms:*

$$\begin{aligned} f(x_1, \ldots, x_k) &= h(p_1(x_1) + \cdots + p_k(x_k)) \\ f(x_1, \ldots, x_k) &= h(p_1(x_1) \cdot \ldots \cdot p_k(x_k)) \end{aligned} \tag{5}$$

Note that the bound in Theorem 1, for non-special polynomials f, is independent of k, and in particular coincides with the bound for $k = 3$. At first glance this may appear to be merely a consequence of the proof. Indeed, the argument in [10] reduces the k-variate case for $k \geq 4$ to the trivariate case by fixing values for $k-3$ of the variables. They then show that if fixing any such subset of $k-3$ variables yields a special trivariate polynomial, then f itself, as a k-variate polynomial, must be special in the sense of (5).

It is natural to expect that increasing the number of variables should force the image of f to grow faster. However, certain polynomials in many variables can in effect behave like polynomials in fewer variables. For example, consider the $(k+2)$-variate polynomial

$$f(x, y, z_1 \ldots, z_k) = xy + z_1 + z_2 + \cdots + z_k.$$

Let $A, B, C_1, \ldots, C_k \subset \mathbb{R}$, where A, B are arbitrary finite sets of size n and $C_1 = \cdots = C_k = [n]$. Let

$$C := C_1 + \cdots + C_k = \{k, k+1, \ldots, kn\}.$$

Then $|C| = \Theta(n)$, and letting $g(x, y, z) := xy + z$, we have

$$f(A, B, C_1, \ldots, C_k) = g(A, B, C).$$

In this case, with the current techniques, it is unclear how to obtain a bound on the expansion of f that improves upon the trivariate result for g.

Our results

In this paper, we recognize k-variate polynomials that are, in a precise sense, truly k-variate, and we improve the corresponding expansion bounds for them. More precisely, for a k-variate polynomial f, we introduce the notion of the *rank* of f. If f has rank r, then, in a rigorous sense, it is essentially $(r+1)$-variate, and the bound on the size of its image can be improved with an exponent that grows with r.

We now define the rank of a polynomial and then state our main result.

Let $f \in \mathbb{R}[x_1, \ldots, x_k]$ and let d_{x_1} stand for the degree of f with respect to the variable x_1. Write

$$f(x_1, \ldots, x_k) = \sum_{i=0}^{d_{x_1}} \alpha_i(x_2, \ldots, x_k)\, x_1^i.$$

We consider the *coefficient map* $T = T_{f,x_1} : \mathbb{R}^{k-1} \to \mathbb{R}^{d_{x_1}+1}$ given by

$$(x_2, \ldots, x_k) \mapsto \left(\alpha_0(x_2, \ldots, x_k), \ldots, \alpha_{d_{x_1}}(x_2, \ldots, x_k)\right).$$

We define the rank of f with respect to the variable x_1 to be

$$\operatorname{rank}_{x_1}(f) := \operatorname{rank}(J_T),$$

where J_T stands for the Jacobian matrix of T; here we consider the rank over the field of rational functions in $x_2, \ldots, x_k$ over $\mathbb{R}$, which is equivalent to evaluating the rank at a generic point $(x_2, \ldots, x_k) \in \mathbb{R}^{k-1}$. Note that

$$0 \le \operatorname{rank}_{x_1}(f) \le k - 1.$$

Similarly, define $\operatorname{rank}_{x_i}(f)$, for every $i = 2, \ldots, k$, where x_i plays the role of x_1.

Finally, define the *rank* of the polynomial f to be

$$\operatorname{rank}(f) := \max_{1 \le i \le k} \operatorname{rank}_{x_i}(f).$$

▶ **Example 2.** Let

$$f(x_1, x_2, \ldots, x_k) = x_1 x_k + x_2 x_k^2 \cdots + x_{k-1} x_k^{k-1}.$$

Then $\operatorname{rank}(f) = \operatorname{rank}_{x_k}(f) = k - 1$.

▶ **Example 3.** Let

$$f(x_1, x_2, \ldots, x_k) = p_1(x_1)x_k + p_2(x_1, x_2)x_k^2 + \cdots + p_{k-1}(x_1, \ldots, x_{k-1})x_k^{k-1},$$

where p_i is an i-variate polynomial that depends non-trivially on x_i. Then $\operatorname{rank}(f) = \operatorname{rank}_{x_k}(f) = k - 1$. Indeed, in this case the matrix $J_{T_{f,x_k}}$ is upper-triangular.

▶ **Example 4.** Let $f(x_1, \ldots, x_k) = p_1(x_1) + p_2(x_2) + \cdots + p_k(x_k)$. The coefficient map T_{f,x_1} maps $(x_2, \ldots, x_k)$ to the coefficients of $p_1(x_1)$, which consist only of a single term depending on the other variables. The Jacobian matrix consists almost entirely of zeros, yielding $\operatorname{rank}(f) = 1$. This aligns perfectly with the special degenerate forms described in Theorem 1.

We prove the following main result of the paper.

▶ **Theorem 5.** *Let $k \ge 3$ and let $f \in \mathbb{R}[x_1 \ldots, x_k]$. Assume that $\operatorname{rank}(f) = r \ge 2$. Then, for every $\varepsilon > 0$, the following holds: Let $A_1, \ldots, A_k \subseteq \mathbb{R}$ be finite, each of size n. Then*

$$|f(A_1, \ldots, A_k)| = \Omega\left(n^{\frac{5r-4}{2r} - \varepsilon}\right),$$

where the constant of proportionality depends on $\deg(f)$, on r, and on ε.

Observe that for a generic k-variate polynomial, the rank is maximal, meaning $r = k-1$. In this case, Theorem 5 yields a lower bound of $\Omega(n^{\frac{5k-9}{2k-2}-\epsilon})$. Notably, this relates to Conjecture 1.3 of de Zeeuw [2], which posits that a generic bivariate polynomial has at least $\Omega(n^{2-\epsilon})$ distinct values. Our theorem pushes towards this super-quadratic growth and currently provides the best known lower bound for the minimal number of distinct values taken by generic k-variate polynomials for $k \geq 4$.

We observe that $\operatorname{rank}(f) = 1$ corresponds to the special forms from Theorem 1. Indeed, we have the following theorem.

▶ **Theorem 6.** *Let $k \geq 3$ and let $f \in \mathbb{R}[x_1 \ldots, x_k]$. Assume that f depends non-trivially on each of its variables and that*

$$\operatorname{rank}(f) = 1.$$

Then f has one of the forms

$$\begin{aligned} &f(x_1, \ldots, x_k) = h(p_1(x_1) + \cdots + p_k(x_k)) \quad \text{or} \\ &f(x_1, \ldots, x_k) = h(p_1(x_1) \cdot \ldots \cdot p_k(x_k)), \end{aligned} \tag{6}$$

for some univariate real polynomials $h(x), p_1(x), \ldots, p_k(x)$.

Finally, we present an application of our results to the following Erdős-type combinatorial geometric problem. Let ν denote the moment curve in $\mathbb{R}^d$, parameterized by

$$\nu(t) = (t, t^2, \ldots, t^d), \quad t \in \mathbb{R}.$$

Let $P \subset \nu$ be a finite set of n points. For any distinct $p_1, \ldots, p_{d+1} \in \nu$, let $\sigma = \sigma(p_1, \ldots, p_{d+1})$ denote the d-simplex which is the convex hull of $p_1, \ldots, p_{d+1}$ in $\mathbb{R}^d$, and let $\operatorname{vol}(\sigma)$ denote its d-dimensional volume. Define

$$\Delta(P) = \{\operatorname{vol}(\sigma(p_1, \ldots, p_{d+1})) \mid p_1, \ldots, p_{d+1} \in P\}.$$

We have the following theorem.

▶ **Theorem 7.** *Let ν be the moment curve in $\mathbb{R}^d$ and let $P \subset \nu$ be any finite set of size n. Then, for every $\varepsilon > 0$,*

$$|\Delta(P)| = \Omega\left(n^{\frac{5d-4}{2d}-\varepsilon}\right),$$

where the implicit constant depends only on ε and on d.

Theorem 7 is obtained by identifying a $(d+1)$-variate polynomial f whose expansion over a certain $n \times \cdots \times n$ grid in $\mathbb{R}^{d+1}$ corresponds to the number of distinct volumes of d-simplices spanned by P. We then show that f has rank d and apply our main Theorem 5.

Organization of the paper

The paper is organized as follows. In Section 2 we recall an incidence bound that will serve as a key tool in our arguments. In Section 3, we establish a special case of our main result, Theorem 5, and in Section 4, we complete its proof. The proof of Theorem 6 is provided in Section 5. Finally, Section 6 contains the proof of Theorem 7.

2 Incidences between points and algebraic curves

For a finite set of points $\mathcal{P} \subset \mathbb{R}^2$ and a finite set of planar curves $\mathcal{C}$, we let $I(\mathcal{P},\mathcal{C})$ denote the set of point-curve incidences; that is

$$I(\mathcal{P},\mathcal{C}) = \{(p,\gamma) \in \mathcal{P} \times \mathcal{C} \mid p \in \gamma\}.$$

The classical Szemerédi–Trotter theorem [14] asserts that, for the special case where $\mathcal{C}$ is a set of lines, and putting $m := |\mathcal{P}|$ and $n := |\mathcal{C}|$, one has

$$|I(\mathcal{P},\mathcal{C})| = O\left(m^{2/3}n^{2/3} + m + n\right).$$

Since the Szemerédi–Trotter result, many alternative proofs and analogue problems have been studied. Today incidence problems play a fundamental role in combinatorial geometry. For our result we will need an extension of the Szemerédi–Trotter theorem to point-curve incidence problems, where the curves are algebraic and come from an s-dimensional family of curves. We now present the definition from Sharir–Zahl [12].

A bivariate polynomial $h \in \mathbb{R}[x,y]$ of degree at most D is a linear combination of the form

$$h(x,y) = \sum_{0 \le i+j \le D} c_{ij} x^i y^j.$$

Note that the number of monomials $x^i y^j$ such that $0 \le i+j \le D$ is $\binom{D+2}{2}$. In this sense, every point $\vec{c} \in \mathbb{R}^{\binom{D+2}{2}}$ (other than the all-zero vector) can be associated with a curve in $\mathbb{R}^2$, given by the zeroset of the bivariate polynomial whose coefficients are the entries of $\vec{c}$. If $\lambda \neq 0$, then f and λf have the same zero-set. Thus, the set of algebraic curves that can be defined by a polynomial of degree at most D in $\mathbb{R}^2$ can be identified with the points in the projective space $\mathbf{P}\mathbb{R}^{\binom{D+2}{2}}$.

In [12], Sharir and Zahl defined an *s-dimensional family of plane curves of degree at most D* to be an algebraic variety $F \subset \mathbf{P}R^{\binom{D+2}{2}}$ such that $\dim(F) = s$. We will call the degree of the variety F the *complexity of the family*.

They then proved the following incidence bound:

▶ **Theorem 8** (Sharir–Zahl [12]). *Let $\mathcal{F}$ be an s-dimensional family of plane curves of degree at most D and complexity at most K. Let $\mathcal{P}$ be a set of m points in the plane and let $\mathcal{C} \subset \mathcal{F}$ be a set of n plane curves. Suppose that no two of the curves in $\mathcal{C}$ share a common irreducible component. Then for every $\varepsilon > 0$, we have*

$$I(\mathcal{P},\mathcal{C}) = O_\varepsilon\left(m^{\frac{2s}{5s-4}} n^{\frac{5s-6}{5s-4}+\varepsilon}\right) + O\left(m^{2/3}n^{2/3} + m + n\right),$$

where the constant of proportionality depends on s, K, D and in the first term also on ε.

3 A special case

In this section we prove the special case of Theorem 5 with $r = k-1$. This will serve as a key ingredient for the proof of the general result.

▶ **Proposition 9.** *Let $k \ge 3$, $f \in \mathbb{R}[x_1 \ldots, x_k]$, and assume that* $\operatorname{rank}(f) = k-1$. *Then, for every $\varepsilon > 0$, the following holds: Let $A_1, \ldots, A_k \subseteq \mathbb{R}$ be finite, each of size n. Then*

$$|f(A_1, \ldots, A_k)| = \Omega\left(n^{\frac{5(k-1)-4}{2(k-1)}-\varepsilon}\right),$$

where the constant of proportionality depends on $\deg(f)$, *on k, and on ε.*

Proof. Let $f, A_1, \ldots, A_k$ be as in the statement. Up to renaming of the variables, we may assume without loss of generality that $\operatorname{rank}_{x_1}(f) = k-1$. Put

$$B = f(A_1, \ldots, A_k)$$

We aim to lower bound $|B|$. For this, consider:

$$S = \{(x_1, x_2, \ldots, x_k, y) \in A_1 \times A_2 \times \cdots \times A_k \times B \mid y = f(x_1, \ldots, x_k)\}.$$

Notice that, for every $(a_1 \ldots, a_k) \in A_1 \times \ldots \times A_k$, we have that $(a_1 \ldots, a_k, f(a_1, \ldots, a_k)) \in S$, and so

$$|S| = n^k. \tag{7}$$

We claim that, for every $\varepsilon > 0$, one has

$$|S| = O\left(|B|^{\frac{2(k-1)}{5(k-1)-4}} n^{(k-1)(1+\varepsilon)}\right), \tag{8}$$

where the constant of proportionality depends on ε, k, and $\deg(f)$. Combining (7) and (8), we get

$$n^k = O\left(|B|^{\frac{2(k-1)}{5(k-1)-4}} n^{(k-1)(1+\varepsilon)}\right)$$

or

$$|B| = \Omega\left(n^{\frac{5(k-1)-4}{2(k-1)} - \varepsilon'}\right),$$

where $\varepsilon' = \frac{5(k-1)-4}{2}\varepsilon$, which proves the proposition.

So in order to complete the proof of Proposition 9 we only need to prove (8). Let $T = T_{f,x_1}$ be the coefficient map defined in the introduction. By assumption, $\operatorname{rank}(J_T) = k-1$. Thus, there exist indices $(i_1, \ldots, i_{k-1})$ such that for

$$\hat{T} : (x_2 \ldots, x_k) \mapsto (\alpha_{i_1}(x_2 \ldots, x_k), \ldots, \alpha_{i_{k-1}}(x_2 \ldots, x_k)),$$

we have

$$\det J_{\hat{T}} \not\equiv 0.$$

Define

$$S_0 = \left\{(a_1, \ldots, a_k, b) \in S \mid \det J_{\hat{T}}(a_2, \ldots, a_k) = 0\right\},$$
$$S' = S \setminus S_0.$$

Clearly $|S| = |S_0| + |S'|$.

Observe that,

$$|S_0| = |A_1| \cdot \left|(A_2 \times \cdots \times A_k) \cap \left\{\det J_{\hat{T}} = 0\right\}\right| \leq n \cdot \deg(\det J_{\hat{T}}) n^{k-2},$$

where the inequality is due to the Schwartz–Zippel Lemma (see [11, 16]). Thus, we get

$$|S_0| = O(n^{k-1}), \tag{9}$$

where the constant of proportionality depends on $\deg(f)$ and on k.

We now bound $|S'|$. For this, we reduce the problem into a point-curve incidence problem in the plane as follows. With each $(a_2, \ldots, a_k) \in A_2 \times \cdots \times A_k$ for which $\det J_{\hat{T}}(a_2, \ldots, a_k) \neq 0$, we associate a curve $\gamma_{a_2,\ldots,a_k}$ in $\mathbb{R}^2$ given by the equation

$$y = f(x, a_2, \ldots, a_k).$$

Note that $\gamma_{a_2,\ldots,a_k}$ is irreducible for every $(a_2, \ldots, a_k) \in A_2 \times \cdots \times A_k$, because it is defined by an equation of the form $y = P(x)$, making it the graph of a polynomial. Let

$$\mathcal{P} = A_1 \times B,$$
$$\mathcal{C} = \left\{\gamma_{a_2,\ldots,a_k} \mid \det J_{\hat{T}}(a_2, \ldots, a_k) \neq 0\right\};$$

note that curves in $\mathcal{C}$ are taken without multiplicity. Let $I(\mathcal{P}, \mathcal{C})$ denote the set of point-curve incidences between $\mathcal{P}$ and $\mathcal{C}$.

▷ Claim 10. We have

$$|S'| = \Theta\left(|I(\mathcal{P}, \mathcal{C})|\right), \tag{10}$$

where the constant of proportionality depends only on $\deg f$ and on k.

Proof. By definition, if $(a_1, a_2, \ldots, a_k, b) \in S'$ then $((a_1, b), \gamma_{a_2,\ldots,a_k}) \in I(\mathcal{P}, \mathcal{C})$. So, to prove the claim, it suffices to show that every $(p, \gamma) \in I(\mathcal{P}, \mathcal{C})$ corresponds to at most $O(1)$ elements of S'.

For $\gamma \in \mathcal{C}$, write

$$m(\gamma) = \left\{(a_2, \ldots, a_k) \mid \gamma_{a_2,\ldots,a_k} = \gamma \text{ and } \det J_{\hat{T}}(a_2, \ldots, a_k) \neq 0\right\}.$$

We need to prove that

$$|m(\gamma)| = O(1), \tag{11}$$

with constant of proportionality that depends only on $\deg(f)$ and on k.

By the definition of the set $\mathcal{C}$, we have that γ is given by an equation of the form

$$y = \sum_{i=0}^{d_{x_1}} c_i x^i,$$

for some coefficients $c_0, \ldots, c_{d_{x_1}} \in \mathbb{R}$. Let V denote the algebraic variety which is given by the system of equations

$$\begin{aligned} \alpha_{i_1}(x_2, \ldots, x_k) &= c_{i_1} \\ \alpha_{i_2}(x_2, \ldots, x_k) &= c_{i_2} \\ &\vdots \\ \alpha_{i_{k-1}}(x_2, \ldots, x_k) &= c_{i_{k-1}}. \end{aligned} \tag{12}$$

Then

$$m(\gamma) \subset V.$$

Write $V = V_0 \cup V_+$, where V_0 is the union of all 0-dimensional irreducible components of V, and V_+ is the union of all other irreducible components of V. Recall that, by properties of real algebraic varieties, we have that V_0 is finite, and

$$|V_0| = O(1),$$

with a constant that depends only of $\deg(f)$ and on k (see e.g. [1, 6, 15] for a bound on the number of connected components of a real algebraic variety).

Thus, to prove (11), it suffices to show that $m(\gamma) \subset V_0$. Assume, for contradiction, that this is not the case. That is, there exists $(a_2, \ldots, a_k) \in m(\gamma) \cap V_+$. Then, by definition, we have

$$\hat{T}(a_2, \ldots, a_k) = (c_{i_1}, \ldots, c_{i_{k-1}})$$
$$\det J_{\hat{T}}(a_2, \ldots, a_k) \neq 0.$$

By the inverse function theorem, there exists an open neighborhood, N, of $(a_2, \ldots, a_k)$ such that $\hat{T}$ restricted to N is invertible. In particular,

$$N \cap \hat{T}^{-1}\{(c_{i_1}, \ldots, c_{i_{k-1}})\} = \{(a_2, \ldots, a_k)\}.$$

On the other hand,

$$N \cap \hat{T}^{-1}(c_{i_1}, \ldots, c_{i_{k-1}}) = N \cap V.$$

Since N is a neighborhood of $(a_1, \ldots, a_{k-1})$ and the latter lies on an irreducible component of V of dimension at least 1, the intersection $N \cap V$ must be infinite. This leads to a contradiction, and thus $m(\gamma) \subset V_0$. This completes the proof of Claim 10. ◁

In view of Claim 10, in order to bound $|S'|$ it suffices to bound $|I(\mathcal{P}, \mathcal{C})|$. We have $|\mathcal{P}| = n|B|$ and $|\mathcal{C}| \leq n^{k-1}$. Since the curves in $\mathcal{C}$ are irreducible, every two distinct curves γ, γ' in $\mathcal{C}$ intersect in at most $d_{x_1}^2$ points, by Bezout's Theorem (see e.g. [5, Corollary 7.8]). We can therefore apply Theorem 8. In our context, the family of curves $\mathcal{F}$ is parameterized by the $k-1$ variables from $A_2 \times \cdots \times A_k$, so the dimension is $s \leq k-1$. The degree $D \leq \max(\deg(f), 1)$ and the complexity $K = O(1)$ depend strictly on k and $deg(f)$, and crucially, are independent of n. This gives

$$|I(\mathcal{P}, \mathcal{C})| = O\left((|B| \cdot n)^{\frac{2(k-1)}{5(k-1)-4}} n^{(k-1)\left(\frac{5(k-1)-6}{5(k-1)-4}+\varepsilon\right)} + (|B|n)^{2/3}(n^{k-1})^{2/3} + |B|n + n^{k-1} \right).$$

or

$$|I(\mathcal{P}, \mathcal{C})| = O\left(|B|^{\frac{2(k-1)}{5(k-1)-4}} n^{(k-1)(1+\varepsilon)} + |B|^{\frac{2}{3}} n^{\frac{2k}{3}} + |B|n + n^{k-1} \right).$$

Note that we may assume without loss of generality that the first summand is dominant. Indeed, the second summand is dominant if

$$|B|^{\frac{2(k-1)}{5(k-1)-4}} n^{(k-1)(1+\varepsilon)} \leq |B|^{\frac{2}{3}} n^{\frac{2k}{3}}$$

or

$$|B| \geq n^{\frac{5(k-1)-4}{4}+O(\varepsilon)},$$

which is stronger than the lower bound we wish to prove on $|B|$, for every $k \geq 3$. Similarly, the third summand is dominant if

$$|B|^{\frac{2r}{5r-4}} n^{r(1+\varepsilon)} \leq |B|n$$

or

$$|B| \geq n^{\frac{(k-2)(5(k-1)-4)}{3(k-1)-4}+\varepsilon\frac{(k-1)(5(k-1)-4)}{3(k-1)-4}},$$

which is better than the lower we want to prove on $|B|$ for every k, as is easy to verify. Finally, the fourth summand is always subsumed by the first one.

Hence, either the conclusion of Proposition 9 holds, or we obtain

$$|I(\mathcal{P},\mathcal{C})| = O\left(|B|^{\frac{2(k-1)}{5(k-1)-4}} n^{(k-1)(1+\varepsilon)}\right).$$

In view of Claim 10 and combined with (9), the inequality (8) follows. This completes the proof Proposition 9. ◀

4 Proof of Theorem 5

The following lemma shows that the main Theorem 5 can in fact be reduced to the statement of Proposition 9.

▶ **Lemma 11.** *Let $f \in \mathbb{R}[x_1 \ldots, x_k]$ and assume that $\operatorname{rank}_{x_1}(f) = r < k-1$. Then, up to renaming of the variables $x_2, \ldots, x_k$, we have that*

$$g(x_1, \ldots, x_{r+1}) := f(x_1, x_2, \ldots, x_{r+1}, y_{r+2}, \ldots, y_k)$$

is a $(r+1)$-variate polynomial in $\mathbb{R}[x_1, \ldots, x_{r+1}]$ with $\operatorname{rank}_{x_1}(g) = r$, for all $(y_{r+2}, \ldots, y_k) \in \mathbb{R}^{k-r-1} \setminus Z_0$, where Z_0 is some subvariety of $\mathbb{R}^{k-r-1}$ of codimension at least 1 and degree depending only on k and $\deg(f)$.

Proof. Let f be as in the statement. Let $T = T_{f,x_1}$ be the corresponding coefficient map. By assumption

$$\operatorname{rank}(J_T) = r < k-1. \tag{13}$$

Up to renaming the variables, we may assume without the loss of generality, that the first r columns of J_T, corresponding to the variables $x_2, \ldots, x_{r+1}$, are independent. Observe that the matrix composed of the first r columns of J_T, is in fact the Jacobian matrix of the coefficient map T_{g,x_1}, where g is the $(r+1)$-variate polynomial given by

$$g(x_1, x_2, \ldots, x_{r+1}) := f(x_1, x_2, \ldots, x_{r+1}, y_{r+2}, \ldots, y_k);$$

here $x_2, \ldots, x_{r+1}$ are regarded as constant parameters.

Our goal is to show that for $J_{T_{g,x_1}}$, treated as a matrix with entries in $\mathbb{R}(x_1, \ldots, x_{r+1})$, we have that $\operatorname{rank}(J_{T_{g,x_1}}) = r$, for every generic choice of $(y_{r+2}, \ldots, y_k) \in \mathbb{R}^{k-r-1}$.

Let Δ denote the polynomial corresponding to the sum of squares of the determinants of all the $r \times r$ submatrices of $J_{T_{g,x_1}}$. So Δ is a multivariate polynomial, and we can write

$$\Delta(x_2, \ldots, x_{r+1}, y_{r+2}, \ldots, y_k) = \sum_i \beta_i(y_{r+2}, \ldots, y_k) g_i(x_2, \ldots, x_{r+1}),$$

with $g_i \in \mathbb{R}[x_2, \ldots, x_{r+1}]$ and $\beta_i \in \mathbb{R}[y_{r+2}, \ldots, y_k]$. By the definition of Δ we have

$$\operatorname{rank}(J_{T_{g,x_1}}(x_2, \ldots, x_{r+1}, y_{r+2}, \ldots, y_k)) < r \text{ if and only if } \Delta(x_2, \ldots, x_{r+1}, y_{r+2}, \ldots, y_k) = 0.$$

Thus, in view of (13), we have $\Delta \not\equiv 0$. In particular, the polynomials β_i are not all zero. Thus, letting

$$Z_0 := \{(y_{r+2}, \ldots, y_k) \mid \forall i \ \beta_i(y_{r+2}, \ldots, y_k) = 0\},$$

we see that Z_0 has codimension at least 1. This completes the proof of the lemma. ◀

We can now complete the proof of our main Theorem 5.

Proof of Theorem 5. Let $f \in \mathbb{R}[x_1, \ldots, x_k]$ and assume without loss of generality that $\deg_{x_1}(f) = r \leq k-2$. Apply Lemma 11 to f. Then, up to renaming of the variables, there exists an algebraic variety $Z_0 \subset \mathbb{R}^{k-r-1}$, of codimension at least one, and of degree $O(1)$, such that for every $(y_{r+2}, \ldots, y_k) \in \mathbb{R}^{k-r-1} \setminus Z_0$ we have that the polynomial

$$(x_1, \ldots, x_{r+1}) \mapsto f(x_1, \ldots, x_{r+1}, y_{r+2}, \ldots, y_k)$$

is an $(r+1)$-variate polynomial of rank r.

Observe that, by the Schwartz–Zippel lemma, there exists $(a_{r+2}, \ldots, a_k) \in (A_{r+2} \times \ldots \times A_k) \setminus Z_0$. Thus

$$g(x_1, \ldots, x_{r+1}) := f(x_1, \ldots, x_{r+1}, a_{r+2}, \ldots, a_k)$$

satisfies $\operatorname{rank}_{x_1}(g) = r$. Thus, by Proposition 9, we have that

$$|g(A_1, \ldots, A_{r+1})| = \Omega\left(n^{\frac{5r-4}{2r}-\varepsilon}\right).$$

Noting that

$$g(A_1, \ldots, A_{r+1}) = f(A_1, \ldots, A_{r+1}, \{a_{r+2}\}, \ldots, \{a_k\}) \subset f(A_1, \ldots, A_k),$$

this completes the proof of the theorem. ◀

5 Characterization of rank-1 polynomials

In this section we prove Theorem 6. For the proof we will use the following lemma from Raz and Shem Tov [10].

▶ **Lemma 12** (Raz–Shem Tov [10, Lemma 2.3]). *Let $f \in \mathbb{R}[x_1, \ldots, x_k]$. Assume that*

$$\frac{\frac{\partial f}{\partial x_1}(x_1, \ldots, x_k)}{r_1(x_1)} = \cdots = \frac{\frac{\partial f}{\partial x_k}(x_1, \ldots, x_k)}{r_k(x_k)}, \tag{14}$$

for some univariate real polynomials $r_1, \ldots, r_k$. Then, f is one of the forms

$$f(x_1, \ldots, x_k) = h(p_1(x_1) + \cdots + p_k(x_k)) \quad \text{or}$$
$$f(x_1, \ldots, x_k) = h(p_1(x_1) \cdot \cdots \cdot p_k(x_k)),$$

for some univariate real polynomials $h(x), p_1(x), \ldots, p_k(x)$.

Proof of Theorem 6. Let f be as in the statement. Since f depends non-trivially on each of its variables, we have in particular that

$$\operatorname{rank}_{x_i}(f) = 1, \quad \text{for each } i = 1, \ldots, k. \tag{15}$$

In view of Lemma 12, it suffices to show that f satisfies the differential equation (14), for some univariate polynomials $r_1, \ldots, r_k$. By symmetry, it suffices to prove that

$$\frac{\frac{\partial f}{\partial x_1}(x_1, \ldots, x_k)}{r_1(x_1)} = \frac{\frac{\partial f}{\partial x_2}(x_1, \ldots, x_k)}{r_2(x_2)} \tag{16}$$

We write

$$f(x_1, \ldots, x_k) = \sum_{i=0}^{d_{x_1}} \alpha_i^{(1)}(x_2, \ldots, x_k)\, x_1^i = \cdots = \sum_{i=0}^{d_{x_k}} \alpha_i^{(k)}(x_1, \ldots, x_{k-1})\, x_k^i, \tag{17}$$

where d_{x_j} stands for the degree of f as a univariate polynomial in the variable x_j. Then for every $j, \ell \in \{1, \ldots, k\}$, such that $j \neq \ell$, we have

$$\frac{\partial f}{\partial x_j}(x_1, \ldots, x_k) = \sum_{i=0}^{d_{x_\ell}} \frac{\partial \alpha_i^{(\ell)}(x_1, \ldots, \widehat{x_\ell}, \ldots, x_k)}{\partial x_j} x_\ell^i. \tag{18}$$

We next show that

$$\frac{\frac{\partial f}{\partial x_1}}{\frac{\partial f}{\partial x_2}} = u(x_1, x_2), \tag{19}$$

where u is some rational function over $\mathbb{R}$.

In other words, we need to show that $\frac{\partial f}{\partial x_1} / \frac{\partial f}{\partial x_2}$ is independent of x_i, for every $i \neq 1, 2$. By symmetry it suffices to show that this ratio is independent of the variable x_3. Consider the first two columns of $J_{T_{f,x_3}}$, corresponding to derivatives with respect to x_1 and to x_2. Namely,

$$\begin{pmatrix} \frac{\partial \alpha_0^{(3)}}{\partial x_1} & \frac{\partial \alpha_0^{(3)}}{\partial x_2} \\ \frac{\partial \alpha_1^{(3)}}{\partial x_1} & \frac{\partial \alpha_1^{(3)}}{\partial x_2} \\ \vdots & \vdots \\ \frac{\partial \alpha_{d_{x_3}}^{(3)}}{\partial x_1} & \frac{\partial \alpha_{d_{x_3}}^{(3)}}{\partial x_2} \end{pmatrix}.$$

Note that neither of the columns is zero, since f depends non-trivially in each of its variables. Moreover, since $\operatorname{rank}(J_{T_{f,x_3}}) = 1$, by (15), and using the fact that the entries of $J_{T_{f,x_3}}$ are independent of x_3, we get that there exists a rational function $u(x_1, x_2, x_4, \ldots, x_k)$ such that

$$\frac{\partial \alpha_i^{(3)}}{\partial x_1} = u(x_1, x_2, x_4, \ldots, x_k) \frac{\partial \alpha_i^{(3)}}{\partial x_2}, \quad \text{for } i = 0, \ldots, d_{x_3}.$$

Using the identity (18), this implies that

$$\frac{\frac{\partial f}{\partial x_1}}{\frac{\partial f}{\partial x_2}} = u(x_1, x_2, x_4, \ldots, x_k);$$

that is, this ratio is independent of x_3. This proves (19).

Repeating the analysis symmetrically for the pair x_1, x_3 and then to the pair x_2, x_3 we conclude that there exist real rational functions $v(x_1, x_3)$ and $w(x_2, x_3)$ such that

$$\frac{\frac{\partial f}{\partial x_1}}{\frac{\partial f}{\partial x_3}} = v(x_1, x_3)$$

$$\frac{\frac{\partial f}{\partial x_2}}{\frac{\partial f}{\partial x_3}} = w(x_2, x_3).$$

But then

$$\frac{\frac{\partial f}{\partial x_1}}{\frac{\partial f}{\partial x_2}} = \frac{v(x_1, x_3)}{w(x_2, x_3)} = u(x_1, x_2),$$

meaning in particular that the ratio of v, w is independent of x_3. Thus, setting an arbitrary value for x_3, we get that

$$\frac{\frac{\partial f}{\partial x_1}}{\frac{\partial f}{\partial x_2}} = \frac{v(x_1, 0)}{w(x_2, 0)}.$$

Thus, letting $r_1(x_1) := v(x_1, 0)$ and $r_2(x_2) := w(x_2, 0)$, this proves (16), and hence completes the proof of the lemma. ◀

6 Distinct d-volumes on the moment curve in $\mathbb{R}^d$

In this section we prove Theorem 7.

Proof of Theorem 7. The *moment curve* in $\mathbb{R}^d$ is defined by $\nu(x) = (x, x^2, \ldots, x^d)$. Consider $d+1$ point on ν given by the parameters $x_1, \ldots, x_{d+1} \in \mathbb{R}^d$, and let $\sigma(x_1, \ldots, x_{d+1})$ denote the simplex spanned by these points. Define

$$f(x_1, \ldots, x_{d+1}) := \mathrm{vol}\sigma(x_1, \ldots, x_{d+1}).$$

It suffices to prove that

$$\mathrm{rank}_{x_{d+1}}(f) = d. \tag{20}$$

Indeed, assume that (20) is true and let $P \subset \nu$ be a set of size n. Let A denote the x_1-coordinate of the points in P. Then $\Delta(P) = f(A, \ldots, A)$. By Theorem 5, for every $\varepsilon > 0$, we have

$$|\Delta(P)| = |f(A, \ldots, A)| = \Omega\left(n^{\frac{5d-4}{2d} - \varepsilon}\right),$$

as needed.

So we only need to prove (20). We have

$$f(x_1, \ldots, x_{d+1}) = \frac{1}{d!} \det \begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^d \\ 1 & x_2 & x_2^2 & \ldots & x_2^d \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{d+1} & x_{d+1}^2 & \ldots & x_{d+1}^d \end{pmatrix},$$

which is the determinant of the $(d+1) \times (d+1)$-*Vandermonde matrix*. Thus

$$\begin{aligned} f(x_1, \ldots, x_{d+1}) &= \frac{1}{d!} \prod_{1 \le i < j \le d+1} (x_j - x_i) \\ &= \frac{1}{d!} \prod_{1 \le i < j \le d} (x_j - x_i) \prod_{k=1}^{d} (x_{d+1} - x_k) \\ &= g(x_1, \ldots, x_d) \hat{f}(x_1, \ldots, x_{d+1}), \end{aligned}$$

where $g(x_1, \ldots, x_d) := \frac{1}{d!} \prod_{1 \le i < j \le d} (x_j - x_i)$ is independent of x_{d+1} and $\hat{f}(x_1, \ldots, x_{d+1}) := \prod_{k=1}^{d} (x_{d+1} - x_k)$. Note that $\hat{f}$ can be written as

$$\hat{f}(x_1, \ldots, x_{d+1}) = \sum_{\ell=0}^{d} s_\ell(x_1, \ldots, x_d) x_{d+1}^\ell,$$

where

$$\begin{aligned} s_0(x_1, \ldots, x_d) &= (-1)^d x_1 \cdots x_d \\ s_1(x_1, \ldots, x_d) &= (-1)^{d-1} (x_2 \cdots x_d + x_1 x_3 \cdots x_d + \cdots + x_1 \cdots x_{d-1}) \\ &\ldots \\ s_{d-1}(x_1, \ldots, x_d) &= -x_1 - \cdots - x_d \\ s_d(x_1, \ldots, x_d) &= 1, \end{aligned}$$

are the symmetric polynomials.

▷ Claim 13. Let M be the $d \times d$ matrix $M = (M_{i,j})_{0 \le i \le d-1,\ 1 \le j \le d}$ given by

$$M_{i,j} = \frac{\partial s_i}{\partial x_j} \qquad (0 \le i \le d-1,\ 1 \le j \le d).$$

Then $\operatorname{rank}(M) = d$.

The strategy for this proof is adapted from the proof of [[7], section 4].

Proof. We prove that $\det M$ is not the zero polynomial. Consider the function

$$P(x_1, \dots, x_d, t) = \prod_{k=1}^{d} (t - x_k) = \sum_{k=0}^{d} s_k\, t^k,$$

which is a monic polynomial in t.

For $1 \le j \le d$ we have

$$\frac{\partial P}{\partial x_j}(t) = -\prod_{\substack{k=1 \\ k \ne j}}^{d} (t - x_k) = \sum_{k=0}^{d-1} \frac{\partial s_k}{\partial x_j}\, t^k$$

Evaluating at $t = x_i$ yields

$$\frac{\partial P}{\partial x_j}(x_i) = \sum_{k=0}^{d-1} \frac{\partial s_k}{\partial x_j}\, x_i^k = \begin{cases} 0, & i \ne j, \\ -\prod_{k \ne j} (x_j - x_k), & i = j. \end{cases} \tag{21}$$

The identity (21) implies

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{d-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{d-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_d & x_d^2 & \dots & x_d^{d-1} \end{pmatrix} M = \operatorname{diag}\left(\frac{\partial P}{\partial x_1}(x_1), \dots, \frac{\partial P}{\partial x_d}(x_d) \right).$$

Thus

$$\begin{aligned} \det(M) &= \frac{(-1)^d \prod_{j=1}^{d} \prod_{i \ne j} (x_j - x_i)}{\prod_{1 \le i < j \le d} (x_j - x_i)} \\ &= \frac{(-1)^d \prod_{1 \le i \ne j \le d} (x_j - x_i)}{\prod_{1 \le i < j \le d} (x_j - x_i)} \\ &= \prod_{1 \le i < j \le d} (x_j - x_i). \end{aligned}$$

This completes the proof of Claim 13. ◁

We are now ready to prove (20). Write

$$f(x_1, \dots, x_{d+1}) = \sum_{i=0}^{d} \alpha_i(x_1, \dots, x_d) x_{d+1}^i.$$

By the above we have

$$\alpha_i = g(x_1, \dots, x_d) s_i(x_1, \dots, x_d), \text{ for } i = 0, \dots, d$$

Let $T = T_{f,x_{d+1}}$. Denoting partial derivatives with subscripts (e.g., $g_{x_1} = \frac{\partial g}{\partial x_1}$), we have

$$J_T = \begin{pmatrix} g_{x_1}s_0 + gs_{0x_1} & g_{x_2}s_0 + gs_{0x_2} & \cdots & g_{x_d}s_0 + gs_{0x_d} \\ \vdots & \vdots & & \vdots \\ g_{x_1}s_{d-1} + gs_{d-1x_1} & g_{x_2}s_{d-1} + gs_{d-1x_2} & \cdots & g_{x_d}s_{d-1} + gs_{d-1x_{d-1}} \\ g_{x_1} & g_{x_2} & \cdots & g_{x_d} \end{pmatrix}.$$

Applying row operation to the first d rows, namely, $R_i \leftarrow R_i - s_i R_d$ we get

$$\begin{aligned} \operatorname{rank} J_T &= \operatorname{rank} \begin{pmatrix} gs_{0x_1} & gs_{0x_2} & \cdots & gs_{0x_d} \\ \vdots & \vdots & & \vdots \\ gs_{d-1x_1} & gs_{d-1x_2} & \cdots & gs_{d-1x_{d-1}} \\ g_{x_1} & g_{x_2} & \cdots & g_{x_d} \end{pmatrix} \\ &= \operatorname{rank} \begin{pmatrix} s_{0x_1} & s_{0x_2} & \cdots & s_{0x_d} \\ \vdots & \vdots & & \vdots \\ s_{d-1x_1} & s_{d-1x_2} & \cdots & s_{d-1x_{d-1}} \\ g_{x_1}/g & g_{x_2}/g & \cdots & g_{x_d}/g \end{pmatrix}. \end{aligned}$$

Note that first d rows of the above matrix are exactly the matrix M from Claim 13. This completes the proof of Theorem 7 ◀

References

1. S. Basu, R. Pollack, and M.F. Coste-Roy. *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics. Springer Berlin Heidelberg, 2007. URL: https://books.google.co.il/books?id=8Hyp1SLis4IC.
2. Frank de Zeeuw. A survey of elekes-rónyai-type problems. *arXiv: Combinatorics*, pages 95–124, 2018. URL: https://api.semanticscholar.org/CorpusID:73602404.
3. György Elekes. On linear combinatorics i. concurrency—an algebraic approach. *Combinatorica*, 17(4):447–458, 1997. doi:10.1007/BF01194999.
4. György Elekes and Lajos Rónyai. A combinatorial problem on polynomials and rational functions. *J. Comb. Theory A*, 89(1):1–20, 2000. doi:10.1006/jcta.1999.2976.
5. Robin Hartshorne. *Algebraic Geometry*, volume 52 of *Graduate texts in mathematics*. Springer, 1977. doi:10.1007/978-1-4757-3849-0.
6. J. Milnor. On the betti numbers of real varieties. *Proceedings of the American Mathematical Society*, 15(2):275–280, 1964. URL: http://www.jstor.org/stable/2034050.
7. João F. Queiró. An inverse problem for singular values and the jacobian of the elementary symmetric functions. *Linear Algebra and its Applications*, 197–198:277–282, 1994. doi:10.1016/0024-3795(94)90491-X.
8. Orit E. Raz, Micha Sharir, and Frank De Zeeuw. Polynomials vanishing on cartesian products: the elekes–szabó theorem revisited. *Duke Mathematical Journal*, 165(18):3517–3566, 2016. doi:10.1215/00127094-3674103.
9. Orit E. Raz, Micha Sharir, and Frank de Zeeuw. The elekes–szabó theorem in four dimensions. *Israel J. Math.*, 227:663–690, 2018.
10. Orit E. Raz and Zvika Shem-Tov. Expanding polynomials: A generalization of the elekes–rónyai theorem to d variables. *Combinatorica*, 40(5):721–748, 2020. doi:10.1007/s00493-020-4041-0.
11. Jacob Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27:701–717, 1980. doi:10.1145/322217.322225.
12. Micha Sharir and Joshua Zahl. Cutting algebraic curves into pseudo-segments and applications. *J. Combin. Theory Ser. A*, 150:1–35, 2017. doi:10.1016/j.jcta.2017.02.006.

13 József Solymosi and J. Zhal. Improved elekes–szabó type estimates using proximity. *J. Combin. Theory Ser. A*, 201:105813, 2024. doi:10.1016/j.jcta.2023.105813.

14 Endre Szemerédi and William T. Trotter. A combinatorial distinction between the euclidean and projective planes. *European J. Combin.*, 4:385–394, 1983. doi:10.1016/S0195-6698(83)80036-5.

15 R. Thom. Sur l'homologie des variétés algébriques réelles. In *Differential and Combinatorial Topology (A Symposium in Honor of Marston Morse)*, pages 255–265. Princeton Univ. Press, Princeton, N.J., 1965.

16 Richard Zippel. An explicit separation of relativised random polynomial time and relativised deterministic polynomial time. *Inform. Process. Lett.*, 33:207–212, 1989. doi:10.1016/0020-0190(89)90143-9.

Space-Efficient Approximate Spherical Range Counting in High Dimensions

Andreas Kalavas
Carnegie Mellon University, Pittsburgh, PA, USA

Ioannis Psarros
Archimedes, Athena Research Center, Greece

Abstract

We study the following range searching problem in high-dimensional Euclidean spaces: given a finite set $P \subset \mathbb{R}^d$, where each $p \in P$ is assigned a weight w_p, and radius $r > 0$, we need to preprocess P into a data structure such that when a new query point $q \in \mathbb{R}^d$ arrives, the data structure reports the cumulative weight of points of P within Euclidean distance r from q. Solving the problem exactly seems to require space usage that is exponential to the dimension, a phenomenon known as the curse of dimensionality. Thus, we focus on approximate solutions where points up to $(1+\varepsilon)r$ away from q may be taken into account, where $\varepsilon > 0$ is an input parameter known during preprocessing. We build a data structure with near-linear space usage, and query time in $n^{1-\Theta(\varepsilon^4/\log(1/\varepsilon))} + t_q^{\varrho} \cdot n^{1-\varrho}$, for some $\varrho = \Theta(\varepsilon^2)$, where t_q is the number of points of P in the ambiguity zone, i.e., at distance between r and $(1+\varepsilon)r$ from the query q. To the best of our knowledge, this is the first data structure with efficient space usage (subquadratic or near-linear for any $\varepsilon > 0$) and query time that remains sublinear for any sublinear t_q. We supplement our worst-case bounds with a query-driven preprocessing algorithm to build data structures that are well-adapted to the query distribution.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Approximate range counting, partition trees, high dimensions

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.60

Related Version *Full Version*: `https://arxiv.org/abs/2603.12106`

Funding *Andreas Kalavas*: Supported in part by the U.S. National Science Foundation (NSF) Grants ECCS-2145713, CCF-2403194, CCF-2428569, and ECCS-2432545.
Ioannis Psarros: This work has been partially supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

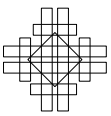
Acknowledgements We would like to thank Dimitris Fotakis for valuable discussions on the topic.

1 Introduction

Range searching is a fundamental problem with wide applications in areas such as computational geometry [13], databases [36], geographic information systems (GIS) [25], and computer graphics [4]. One fundamental variant of range searching is that of *range counting*, where given a dataset P, we need to preprocess it into a data structure so that for a query range R_q, we can efficiently estimate $|R_q \cap P|$. Primary examples of types of query ranges include rectangles, balls, halfspaces, and simplices.

In this paper, we study the special case of range counting queries for Euclidean balls of fixed radius. Solving the problem exactly seems to require space usage that is exponential to the dimension, a phenomenon known as the curse of dimensionality. Therefore, aiming for solutions that scale well with the dimension, we focus on approximate solutions where points up to $(1+\varepsilon)r$ away from q may be considered, where $\varepsilon > 0$ is an input parameter known during preprocessing. Formally, given as input a dataset $P \subseteq \mathbb{R}^d$, such that each point $p \in P$


© Andreas Kalavas and Ioannis Psarros;
licensed under Creative Commons License CC-BY 4.0



is assigned a weight $w_p \in \mathbb{R}$, search radius $r > 0$, and approximation parameter $\varepsilon > 0$, the *approximate spherical range counting* problem is to preprocess P into a data structure that given any query point $q \in \mathbb{R}^d$, it reports $w_S = \sum_{p \in S} w_p$, where $S \subseteq P$ is a subset such that $B(q, r) \cap P \subseteq S \subseteq B(q, (1 + \varepsilon)r) \cap P$, where, for any $x \in \mathbb{R}^d$, $r > 0$, $B(x, r)$ denotes the Euclidean ball of radius r centered at x. Since we can uniformly rescale the dataset, we assume throughout the paper that the search radius is 1.

Related work. We juxtapose the above problem definition with a different notion of approximate counting, adapted for example by Aronov and Har-Peled [10], where one is interested in approximating the number of points inside a query range. Given a query range R_q, the data structure shall return a value ν such that $(1-\varepsilon)|R_q \cap P| \leq \nu \leq (1+\varepsilon)|R_q \cap P|$. Aronov and Har-Peled [10] show that one can reduce such a counting query to a sequence of "emptiness" queries, determining if the query range is empty or not, of length polynomial in $\log |P|$ and ε^{-1}. In contrast, our definition asks for an *exact* aggregate over a set whose geometry is allowed to vary within the annulus $B(q, 1+\varepsilon) \setminus B(q, 1)$. This definition has been used, e.g., by Arya and Mount [11], although they study solutions in the low-dimensional regime.

In terms of known solutions, approximate spherical range counting in high dimensions is not clearly distinguished from other variants of approximate range searching. Perhaps the most promising approach for related problems in high dimensions is locality sensitive hashing (LSH) (see [26, 20, 6, 8]), which maps the points to buckets with the promise that near (resp. far) points are more (resp. less) likely to be mapped together. It is not clear how one could use an LSH-based approach to solve the counting variant without paying at least the cost of exploring all points in $B(q, 1)$. This suggests that, within LSH-based frameworks, counting is unlikely to admit a fundamentally more efficient solution than reporting. In this work, we focus on designing solutions better suited to the counting variant.

Our Results. Our main contribution is to design a data structure for the approximate range counting problem in high-dimensional Euclidean spaces. For any constant approximation parameter $\varepsilon \in (0, 1)$, our data structure uses near-linear space, polynomial preprocessing time, and query time that is sublinear for any sublinear number of points in the ambiguity zone, i.e., for any sublinear $t_q = |\left(B(q, 1+\varepsilon) \setminus B(q, 1)\right) \cap P|$ the query time is sublinear.

Our approach is reminiscent of classic results for the *exact* range searching problem in low-dimensional Euclidean spaces. In particular, a central notion of our approach is that of partition trees [32, 14]. We adapt well-studied relations between spanning trees with low stabbing numbers (which also appears as "crossing" in previous works) and partition trees (see, e.g., [15]) to our approximate setting. A key difference in our setting is that we use a stronger notion of stabbing; we say that a point q ε-stabs a set S if $\exists x \in S : \|x - q\| \leq 1$ and $\exists y \in S : \|y - q\| \geq 1 + \varepsilon$ (classic stabbing would correspond to the case where $\varepsilon = 0$). An important step in our preprocessing algorithm is finding a "light" edge, i.e., a pair of input points that is ε-stabbed by a *polynomial in d* ratio of the relevant queries. Then, using the multiplicative weight update (MWU) method in a similar manner as [38, 15, 39], we are able to show that we can compute a spanning tree with low ε-stabbing number, implying that we can build a partition tree such that for any query, we only need to visit a bounded number of nodes. To traverse the partition tree, we need to be able to decide for each internal node if the corresponding subset is ε-stabbed by the query. We can only do this approximately, meaning that we may also explore nodes that are almost ε-stabbed, but the overall cost of this exploration is proportional to t_q.

▶ **Theorem 1** (Simplified version of Theorem 17). *Given a set of n points in $\mathbb{R}^d$, we can build a randomized data structure for the approximate range counting problem, with space usage in $\tilde{O}(n)$, preprocessing time in $O(dn) + n^{\mathsf{poly}(1/\varepsilon)}$ and sublinear query time if t_q is also sublinear.*

Our data structure is the first to achieve space usage that is near-linear in n and polynomial in d, and query time that is sublinear in n and polynomial in d for any sublinear t_q. In Table 1, we juxtapose our result with previous known results that even implicitly solve the same problem. While LSH is originally proposed for the approximate near neighbor problem, where the data structure is only required to report one witness point within distance $1 + \varepsilon$, Ahle et al. [2] proposed an LSH-based solution for the problem of reporting all points. Their data structure is adaptive; while building time and space can be that of an LSH table prepared for worst-case inputs (parameter ρ in Table 1), query time is that of the best LSH parameters for the query/instance (parameter b). Moreover, LSH tuned for near-linear space [8] can solve the reporting problem with only a logarithmic overhead in space and query time, since we can amplify the probability that every near neighbor will eventually be encountered. Since both LSH-based methods can solve the reporting variant, they can also solve the counting problem. However, it is not clear whether one can modify such an approach to avoid looking into all near points. Although our result does not guarantee sublinear query time in the worst case, it advances prior space-efficient work in high dimensions by making the complexity depend on the number of points in the annulus rather than on all points inside the entire ball. Trivially, the bucketing method, when combined with a Johnson-Lindenstrauss dimension reduction (as in [20]), can solve the counting problem; after reducing the dimension, one can store the relevant information in grid cells near the dataset. Such methods that discretize the query space, while providing efficient query times, unavoidably have huge space requirements. Finally, when the dimension is constant, partition trees [14] solve the problem exactly, while BBD-trees [11] leverage improved query times for approximate solutions (although still exponential in d).

In addition to our worst-case guarantees, we present an algorithm inspired by learning theory. Specifically, this algorithm is query-driven, i.e., it receives a number of query samples and aims to construct the data structure that yields the best expected query runtime for the distribution of the queries. This way, we design a much faster preprocessing algorithm (from $n^{\mathsf{poly}(1/\varepsilon)}$, the complexity falls to $n^{O(1)}$), which gives us similar bounds for the average case.

Our approach is motivated by the viewpoint of analyzing algorithms through the lens of statistical learning (e.g., [19, 12]). While this has been influential in optimization and mechanism design, to the best of our knowledge, a similar treatment for data structures is absent. Although related ideas appear in learned index structures [31], LSH trees whose cutting rules are optimized with respect to the given dataset [7], learning-augmented binary trees with optimized search assuming access to advice from a frequency estimation oracle [30], these works do not directly cast the data structure construction itself as a PAC learning problem. We also remark that the related notions of self-improving [3] and instance-optimal algorithms [1] address different guarantees. A simplified version of our result follows:

▶ **Theorem 2** (Simplified version of Theorem 20). *Given a set of n points in $\mathbb{R}^d$ and a sample of $O(nd \log n)$ queries from a query distribution $\mathcal{D}_Q$, we can compute a partition tree $\tilde{\mathcal{T}}$ with near-optimal expected visiting number for $\mathcal{D}_Q$ in $O(n^3 d \log n)$ time with high probability.*

We believe this is a neat way to use tools from other areas to design efficient algorithms for difficult problems or problems with only existence guarantees, and it would be interesting to see what other data-structure problems might benefit from this viewpoint.

Organization. In Section 2 we present notation and useful known results and prove auxiliary lemmas that will be used later. In Section 3 we analyze a data structure that supports approximate stabbing queries. Subsequently, in Section 4 we prove that a spanning tree with sublinear ε-stabbing number exists and can be computed in polynomial time, which we then use to prove that a partition tree with sublinear query time can be efficiently constructed in Section 5. In Section 6 we present our data-driven algorithm for building a partition tree with near-optimal expected visiting number. Missing proofs can be found in the full version.

Table 1 Existing solutions for the approximate range counting problem. k denotes the number of points within distance $1+\varepsilon$ from the query, whereas $t \leq k$ denotes the number of points at distance between 1 and $1+\varepsilon$ from the query. Adaptive LSH uses a worst-case choice of LSH exponent ρ, but answers queries with an instance-best LSH exponent b.

Method	Space	Prep. time	Query time
LSH [8]	$n^{1+o(1)}$	$n^{1+o(1)}$	$n^{1-\Theta(\varepsilon^2)+o(1)} + kn^{o(1)}$
Adaptive LSH [2]	$\tilde{O}(n^{1+\rho})$	$\tilde{O}(n^{1+\rho})$	$O(k(n/k)^b)$
Bucketing + JL [20]	$n^{\mathrm{poly}(1/\varepsilon)}$	$n^{\mathrm{poly}(1/\varepsilon)}$	$O(\varepsilon^{-2}\log n)$
BBD-Trees [11]	$O(dn)$	$O(dn\log n)$	$O(\log n) + O(1/\varepsilon)^{d-1}$
Partition trees [14]	$O(dn)$	$O(dn)$	$O(n^{1-1/d})$
Theorem 20	$\tilde{O}(n)$	$n^{\mathrm{poly}(1/\varepsilon)}$	$n^{1-\Theta(\varepsilon^4/\log(1/\varepsilon))} + t^{\Theta(\varepsilon^2)} \cdot n^{1-\Theta(\varepsilon^2)}$

2 Preliminaries

We use the $\tilde{O}$ notation to hide the dependence on $\mathrm{poly}(\log n, \log(1/\varepsilon))$. For any $\delta \in \mathbb{R}_{>0}$, let $\mathcal{G}_\delta^d = \{\delta \cdot v \mid v \in \mathbb{Z}^d\}$ be the grid of side length δ. Unless stated otherwise, $\|\cdot\|$ denotes the Euclidean norm $\|\cdot\|_2$. For any $p \in \mathbb{R}^d$ and $r > 0$, $B(p,r)$ denotes the Euclidean ball centered at p, i.e., $B(p,r) = \{x \in \mathbb{R}^d \mid \|x-p\| \leq r\}$.

We say that a set of points P is (ε, r)-*stabbed* by a point p, if and only if there exists a point $x \in P$ such that $\|x-p\| \geq (1+\varepsilon)r$ and there exists a point $y \in P$ such that $\|y-p\| \leq r$. We say that a pair of points x, y (or an edge formed by them) is (ε, r)-stabbed by p, if and only if $\{x, y\}$ is (ε, r)-stabbed by p. For brevity, when $r = 1$, we say that a pair x, y is ε-stabbed by p if and only if it is $(\varepsilon, 1)$-stabbed by p. Given some $P \subset \mathbb{R}^d$ and an edge set G (if G is a graph, we refer to its edges), $\sigma(q, G)$ denotes the *ε-stabbing number* of q on G, i.e., the number of edges in G that are ε-stabbed by q.

A *partition tree* $\mathcal{T}$ built on a set of points P is a tree where each node v is associated with a subset $P_v \subseteq P$, and for each node v with children $u_1, \ldots, u_\ell$, we have $P_v = \bigcup_i P_{u_i}$ and $\forall i \neq j \; P_{u_i} \cap P_{u_j} = \emptyset$. For a partition tree $\mathcal{T}$, and a query q, the *visiting number* of $\mathcal{T}$ w.r.t. q, denoted by $\zeta(q, \mathcal{T})$ is the number of nodes v in $\mathcal{T}$ which are either root of $\mathcal{T}$ or their parent u satisfies one of the following:

- P_u is ε-stabbed by q, or
- $P_u \cap (B(q, 1+\varepsilon) \setminus B(q,1)) \neq \emptyset$ and $P_u \subseteq B(q, 1+\varepsilon)$, or
- $P_u \cap (B(q, 1+\varepsilon) \setminus B(q,1)) \neq \emptyset$ and $P_u \cap B(q,1) = \emptyset$.

Intuitively, the visiting number is an upper-bound estimate of the number of nodes that will be visited to answer a query approximately.

2.1 Range spaces

Fix a countably infinite domain set X. The so-called Vapnik-Chernovenkis dimension [34, 35, 37] (VC dimension) of a set $\mathcal{F}$ of functions from X to $\{0,1\}$, denoted by $VCdim(\mathcal{F})$, is the largest d such that there is a set $S = \{x_1, \ldots, x_d\} \subseteq X$, for which, for any $s \in 2^S$ there

is a function $f \in \mathcal{F}$ such that $p \in s \iff f(p) = 1$. A *range space* is defined as a pair $(X, \mathcal{F})$ where X is the domain and $\mathcal{F}$ is a set of functions from X to $\{0, 1\}$, or equivalently a pair of sets $(X, \mathcal{R})$, where X is the domain and $\mathcal{R}$ is a set of ranges, i.e., subsets of X. The dual range space of a range space defined by sets $(X, \mathcal{R})$ is a range space defined by sets $(\mathcal{R}, \mathcal{X})$, where $\mathcal{X} = \{\{R \in \mathcal{R} \mid x \in R\} \mid x \in X\}$. The dual VC dimension of a range space is the VC dimension of its dual. A set $N \subseteq X$ is an ε-net if $\forall R \in \mathcal{R} : \ |R \cap X| \geq \varepsilon |X| \ \Rightarrow \ N \cap R \neq \emptyset$.

▶ **Theorem 3** (ε-net theorem [24]). *Let $(X, \mathcal{R})$ be a range space of VC-dimension $D < \infty$, where X is a finite set. There exists a constant $C > 0$ such that for any $\varepsilon \in (0, 1)$ if we sample a set of m points (independent picks with uniform probability) from X, where*

$$m \ \geq \ C \frac{1}{\varepsilon} \Big(D \log \tfrac{1}{\varepsilon} \ + \ \log \tfrac{1}{\delta} \Big),$$

then, with probability at least $1 - \delta$, the resulting set is an ε-net for $(X, \mathcal{R})$.

For any $S = (x_1, \ldots, x_m) \in X^m$ let $\mu_f(S) = m^{-1} \cdot \sum_{i=1}^{m} f(x_i)$. For any $\eta > 0, r \geq 0, s \geq 0$, let $d_\eta(r, s) = \frac{|r-s|}{\eta + r + s}$. By properly adjusting the parameters in Theorem 3.1 of [29], we obtain the following corollary. A similar result is obtained in [23].

▶ **Corollary 4.** *Let X be a countably infinite domain. Let $\mathcal{F}$ be a set of functions from X to $\{0, 1\}$ with VC dimension D and let $\mathcal{D}$ be a probability distribution over X. Also, let $\delta, \eta \in (0, 1)$. There is a universal constant c such that if $m \geq \frac{c}{\eta} \cdot \left(D \log \left(\frac{1}{\eta} \right) + \log \left(\frac{1}{\delta} \right) \right)$ then if we sample $S = \{x_1, \ldots, x_m\}$ independently at random according to $\mathcal{D}$, then with probability at least $1 - \delta$, for any $f \in \mathcal{F}$,*

$$|\mu_f(S) - \mathbb{E}_{p \sim \mathcal{D}}[f(p)]| \leq (3/8) \cdot \max\{\mathbb{E}_{p \sim \mathcal{D}}[f(p)], \eta\}.$$

A key requirement in building our data structures is to be able to efficiently sample from a multiset, which can also be updated. A multiset can be seen as a weighted set where weights correspond to multiplicities. In the following lemma, we show that one can maintain a multiset in a data structure that supports sampling and updates on the multiplicities.

▶ **Lemma 5.** *[Multiset sampling data structure] Let $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$, and let $\varepsilon \in (0, 1)$. Let $\Gamma = \{\gamma_1, \ldots, \gamma_n\} \subset \mathbb{N}$ be weights associated with the points of P. There is an algorithm that preprocesses Γ, P into a data structure, in $O(n \log n)$ time, using $O(n \log n)$ space, which supports the following operations:*

1. *sampling of a point $p_i \in P$ with probability $\frac{\gamma_i}{\sum_{i \in [n]} \gamma_i}$ in $O(\log(n))$ time,*
2. *update of the weight γ_i of a point $p_i \in P$, in $O(\log n)$ time.*

Sampling uses $O\left(\log\left(\sum_{i \in [n]} \gamma_i\right)\right)$ random bits. If bit-complexity were taken into account, both preprocessing and updates would incur an additional $O\left(\log\left(\sum_{i \in [n]} \gamma_i\right)\right)$ bit-cost per arithmetic operation.

2.2 Embeddings

We use the following version of the Johnson-Lindenstrauss (JL) lemma [28], as expressed for a single vector and Gaussian matrices.

▶ **Theorem 6** ([27]). *Let $x \in \mathbb{R}^d$ be fixed and let $k \in \mathbb{N}$. Draw a random matrix $A \in \mathbb{R}^{k \times d}$ with i.i.d. entries $A_{ij} \sim \mathcal{N}(0, 1/k)$. Then for any $\varepsilon \in (0, 1)$,*

$$\Pr\left[\left| \|Ax\|_2^2 - \|x\|_2^2 \right| > \varepsilon \|x\|_2^2 \right] \ \leq \ 2 \cdot exp\left(-\frac{\varepsilon^2 k}{8} \right).$$

We also make use of the following result on terminal embedding for Euclidean spaces.

▶ **Theorem 7** ([33]). *Let $T \subset \mathbb{R}^d$ be a finite set of* terminals. *For every $\varepsilon \in (0,1)$ there exists a mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ with $k = O(\varepsilon^{-2} \log |T|)$ such that for all $q \in T$ and all $p \in \mathbb{R}^d$,*

$$(1-\varepsilon)\,\|q-p\|_2 \;\le\; \|f(q)-f(p)\|_2 \;\le\; (1+\varepsilon)\,\|q-p\|_2.$$

For any T, and any $c > 0$, we can compute a mapping f in $\mathrm{poly}(|T|)$ *time such that the above property holds with probability at least $1 - n^{-c}$. Moreover, for any $p \in \mathbb{R}^d$, $f(p)$ can be computed in* $\mathrm{poly}(|T|)$ *time.*

2.3 Model

We assume that our algorithms operate in the Real-RAM model with the additional assumption that the floor function can be computed in constant time. While it is known that this model is strictly more powerful than the classical word-RAM [16], it is often used in the Computational Geometry literature [22]. We stress that we do not use encodings that pack superpolynomial information into a single machine word or real number, and our use of $\lfloor \cdot \rfloor$ is purely for convenience (e.g., discretization and bucketing).

One building block of our algorithms is a data structure known as the dictionary. A *dictionary* is a data structure which stores a set of (key, value) pairs and when presented with a key, either returns the corresponding value, or returns that the key is not stored in the dictionary. In our case, keys are typically vectors in $\mathbb{Z}^d$ and values are pointers to lists. In the Real-RAM model, we can use a balanced binary tree to build a dictionary with n values, using $\tilde{O}(dn)$ preprocessing time and storage and $\tilde{O}(d)$ query time.

3 Approximate stabbing queries

In this section, we design a solution for our main subproblem, that is, the problem of preprocessing a set of points to support queries that ask to determine if a query ε-stabs the given pointset. In particular, the data structure is required to determine if the query ε-stabs the pointset, if all points are near the query, or if all points are far from the query. The resulting data structure returns approximate answers, meaning that it allows for wrong decisions when the pointset is "nearly" ε-stabbed, as formalized in Theorem 10. This is sufficient for our purposes, which are building these data structures in internal nodes of a partition tree to assist in traversing the tree to answer a query.

Our main technical ingredient is a randomized embedding to the Hamming metric, which is implemented using LSH to partition the space into buckets and then applying random binarization of the buckets. A similar argument has been used in Alman et al. [5], where they first embed the points to ℓ_1 and then use an LSH function to partition points into buckets; each bucket is then assigned to a random bit. Repeating the above sufficiently many times gives the desired embedding. We obtain a slightly more direct construction that does not need to embed the points in ℓ_1, but instead uses the LSH family of Datar et al. [18]. The final embedding succeeds in preserving distances with respect to a single scale; distances considered small with respect to a given threshold remain small, and large distances remain large. This is formally stated in the following lemma.

▶ **Lemma 8.** *For any $\delta, \varepsilon \in (0,1)$, there exists a randomized embedding $f : \mathbb{R}^d \to \{0,1\}^{d'}$, and specific $\theta \in [d']$, such that, for any $p, q \in \mathbb{R}^d$, with probability at least $1-\delta$,*

- *if $\|p-q\| \le 1$ then $\Pr[\|f(p)-f(q)\|_1 > \theta] \le e^{-\frac{\varepsilon^2 d'}{19200}}$*
- *if $\|p-q\| \ge (1+\varepsilon)$ then $\Pr[f\|(p)-f(q)\|_1 < \theta + \varepsilon d'] \le e^{-\frac{\varepsilon^2 d'}{19200}}$.*

Moreover, for any $p \in \mathbb{R}^d$, $f(p)$ can be computed in $O(d)$ running time.

We aim to build a data structure that supports approximate "stabbing queries". Given a query point q we need to identify if there exists a pair of points in the input pointset that are (approximately) ε-stabbed by q. Using the embedding of Lemma 8, we map points to keys in $\{0,1\}^{d'}$, and we store them in a dictionary, where each bucket is assigned the list of points mapped to it. We set d' slightly below $\log n$ to balance the number of buckets $2^{d'}$ with the number of arbitrarily distorted distances from q, in a way that both remain sublinear. To answer a query, we search the dictionary for an approximate near neighbor, i.e., a point within distance $1+\varepsilon$, and an approximate far neighbor, i.e., a point at a distance of at least 1. The following lemma indicates that a simple search over the buckets of a single dictionary is sufficient to guarantee that if there is a point within distance 1, we will find a point within distance $1+\varepsilon$ with constant probability, and if there is a point at distance at least $1+\varepsilon$, we will find a point at distance at least 1 with constant probability. We remark that for finding an approximate near neighbor, one could instead use the data structure of Andoni et al. [8] that provides near-linear space with comparable query time. However, we are not aware of a comparable result for computing an approximate far neighbor (which we also need and do).

▶ **Lemma 9.** *Let $P=\{p_1,\ldots,p_n\}\subset\mathbb{R}^d$ and let $\varepsilon\in(0,1)$. We can build a randomized data structure with preprocessing time and space in $\tilde{O}(dn)$ and query time in $\tilde{O}(dn^{1-\frac{\varepsilon^2}{19200(1+\varepsilon^2)}})$, such that for any query $q\in\mathbb{R}^d$, it reports $\mathcal{I}$, $|\mathcal{I}|\leq 2$, as follows:*

- *if $\exists x\in P:\ \|x-q\|\leq 1$ then $\Pr[\exists i\in\mathcal{I}$ such that $\|q-p_i\|\leq 1+\varepsilon]\geq 0.95$,*
- *if $\exists y\in P:\ \|y-q\|\geq 1+\varepsilon$ then $\Pr[\exists j\in\mathcal{I}$ such that $\|q-p_j\|\geq 1]\geq 0.95$,*

where the probabilities are taken over the randomness during preprocessing.

Now using Lemma 9, we can prove the main result of this section, namely, a data structure that determines, with high probability, if a query point ε-stabs the preprocessed pointset, if it is near the entire pointset (the pointset is *covered* by q), or if it is far from the entire pointset (the pointset is *disjoint* from q). The data structure gives approximate answers in the following sense: if all points are within distance $1+\varepsilon$, and there is at least one point at a distance at most 1, it can falsely report that the pointset is ε-stabbed. Similarly, if all points are at a distance larger than 1, and there is at least one point at a distance of at least $1+\varepsilon$, it can falsely report that the pointset is ε-stabbed. If all points lie in the annulus $\{x\in\mathbb{R}^d \mid \|x-q\|\in(1,1+\varepsilon)\}$ then any answer is possible. Additionally, the success probability of the preprocessing is for each query individually, i.e., it is not for all queries simultaneously. Copies of this structure will be used as parts of the final data structure for the approximate spherical range counting problem.

▶ **Theorem 10.** *Let $P\subset\mathbb{R}^d$ be a set of n points and let $\varepsilon\in(0,1)$. We can build a randomized data structure such that for any query $q\in\mathbb{R}^d$, it reports as follows:*

- *if $\exists x\in P:\ \|x-q\|\leq 1$ and $\exists y\in P:\ \|y-q\|\geq 1+\varepsilon$ then it returns "stabbed",*
- *if $\forall y\in P:\ \|y-q\|\leq 1$ then it returns "covered"*
- *if $\forall x\in P:\ \|x-q\|\geq 1+\varepsilon$ then it returns "disjoint",*
- *if $\exists x\in P:\ \|x-q\|\leq 1$ and $\forall y\in P:\ \|y-q\|<1+\varepsilon$ then it can return "stabbed", or "covered", and*
- *if $\exists x\in P:\ \|x-q\|\geq 1+\varepsilon$ and $\forall y\in P:\ \|y-q\|>1$ then it can return "stabbed", or "disjoint".*
- *if $\forall x\in P:\ \|x-q\|\in(1,1+\varepsilon)$ then it can return "stabbed", or "disjoint", or "covered".*

For each query and for any constant $c>0$, the preprocessing of the data structure succeeds with probability $1-n^{-c}$, has $\tilde{O}(dn)$ preprocessing time and space and the query time is $\tilde{O}(dn^{1-\frac{\varepsilon^2}{19200(1+\varepsilon^2)}})$.

Proof. We build $L = c \log n$ independent data structures as in Lemma 9. For any query point $q \in \mathbb{R}^d$, we query the data structures, and we report as follows. If we find a point x such that $\|x - q\| \leq 1 + \varepsilon$ and a point y such that $\|y - q\| \geq 1$ we return "stabbed". Otherwise, if all returned points are within distance $1 + \varepsilon$, we return "covered". Else, we return "disjoint".

If there is a pair x, y in P such that $\|x - q\| \leq 1$ and $\|y - q\| \geq 1 + \varepsilon$ then the probability that all L data structures fail to return a point x' such that $\|x' - q\| \leq 1 + \varepsilon$ or a point y' such that $\|y' - q\| \geq 1$ is by union bound at most $2 \cdot 0.05^L \leq n^{-c}$. Moreover, if all points of P are within distance 1 from q (resp. at a distance of at least $1 + \varepsilon$), then all returned points are within distance $1 + \varepsilon$ from q (resp. at a distance of at least 1), implying that the query algorithm correctly identifies "covered" (resp. "disjoint"). ◀

4 Spanning trees with low ε-stabbing number

The ε-stabbing number of a spanning tree (or of a set of edges in general) is the maximum number of edges of that tree (or that set) that are ε-stabbed by any query range. Chazelle and Welzl showed in [15] a close relationship between the stabbing number of a spanning tree (the stabbing number is the special case of ε-stabbing number where $\varepsilon = 0$) and the maximum number of nodes visited, during a worst-case query, in a corresponding partition tree. Motivated by this, we wish to extend this notion to the ε-stabbing number and to achieve similar results, but without exponential dependency on d as we deal with the approximation scheme. In this section, we show that a spanning tree with sublinear ε-stabbing number exists, and can be computed in polynomial time. Following [15], we use MWU to find a spanning tree with small ε-stabbing number. We remark that there are other alternatives [17, 21] in the literature for computing spanning trees with low stabbing numbers. While it is plausible that they can be modified to support our notion of ε-stabbing, they all share the same bottleneck, i.e., they depend at least linearly on the number of ranges.

Similarly, to [15], we start by showing that for any set of ball queries, we can always find a pair of points that is not ε-stabbed many times.

▶ **Lemma 11.** *Let $P \subset \mathbb{R}^d$ be a set of n points and let Q be a multiset consisting of m points from a universe set $U \subset \mathbb{R}^d$. Let $\varepsilon \in (0, 1)$. There exists a pair of points in P which is not ε-stabbed by more than $O\left(m \cdot \frac{d}{n^\rho}\right)$ points of Q, where $\rho = \Theta(\varepsilon^2 / \log(1/\varepsilon))$.*

If Q is stored in a multiset sampling data structure as in Lemma 5, then for any constant c, we can find such a pair of points in $\mathrm{poly}(n, d, \frac{1}{\varepsilon}) \cdot O(\log |U|)$ time with probability $1 - n^{-c}$.

Proof. If $d \geq n^\rho$, then the statement trivially holds, since $|Q| = m$. Thus, we assume $d < n^\rho$. For any two points x, y, let $S_{x,y}$ be the set of points in $\mathbb{R}^d$ that ε-stab x, y. We define a range space $\mathcal{R} = (\mathbb{R}^d, \{S_{x,y} \mid x, y \in \mathbb{R}^d, x \neq y\})$. Notice that for any point $p \in \mathbb{R}^d$, determining if $p \in S_{x,y}$ reduces to evaluating the signs of the following four polynomials of degree 2 in d variables: $\|p - x\|^2 - 1$, $\|p - x\|^2 - (1 + \varepsilon)^2$, $\|p - y\|^2 - 1$, $\|p - y\|^2 - (1 + \varepsilon)^2$. This implies (see, e.g., [9][Theorem 8.3]) that the VC dimension of $\mathcal{R}$ is $O(d)$. Hence, by Theorem 3, for any $\delta < 1$ and constant $c > 0$, a random sample N_δ of $O\left(\frac{d}{\delta}\left(\log\left(\frac{1}{\delta}\right) + \log n\right)\right)$ points of Q is a δ-net for $\mathcal{R}$ with probability $1 - n^{-c-1}$. Moreover, we can find such a δ-net by randomly sampling from Q, using the assumed multiset sampling data structure of Lemma 5, in $\tilde{O}(\frac{d}{\delta} \log n) \cdot O(\log |U|)$ time.

By Theorem 7, there exists a terminal embedding f to dimension $d' = O(\varepsilon^{-2} \log |N_\delta|)$ such that $\forall p \in N_\delta, q \in \mathbb{R}^d$, $\|f(p) - f(q)\| \in (1 \pm \varepsilon/20)\|p - q\|$. The time needed to compute $f(P)$ is in $O(n) \cdot \mathrm{poly}(|N_\delta|)$ for the probability of success to be at least $1 - n^{-c-1}$.

Now let $x, y \in P$ be ε-stabbed by some $q \in Q$, and further assume wlog that $\|x - q\| \leq 1$ and $\|y - q\| \geq 1 + \varepsilon$. We obtain $\|f(x) - f(q)\| \leq 1 + \varepsilon/20$ and $\|f(y) - f(q)\| \geq (1 + \varepsilon)(1 - \varepsilon/20) \geq 1 + 3\varepsilon/4$. Hence, $f(x), f(y)$ are $(\varepsilon/2, 1 + \varepsilon/20)$-stabbed by $f(q)$. Moreover, by the triangle

inequality, $\|f(x) - f(y)\| \geq \varepsilon/2$. Now consider the grid of side-length $\varepsilon/(4\sqrt{d'})$ in $\mathbb{R}^{d'}$ and let U_B be the set of all grid cells that intersect with at least one ball of radius $1+\varepsilon$ centered at a point of $f(N_\delta)$. There are

$$n_{\varepsilon,d'} = |N_\delta| \cdot O(1/\varepsilon)^{d'} = |N_\delta| \cdot O(1/\varepsilon)^{O(\varepsilon^{-2} \cdot \log|N_\delta|)} = |N_\delta|^{O(\varepsilon^{-2}\log(1/\varepsilon))}$$

such cells, where $|N_\delta| = O\left(\frac{d}{\delta}\left(\log\left(\frac{1}{\delta}\right) + \log n\right)\right)$. Now let $\delta = \frac{d}{n^\rho}$ where $\rho = \Theta(\varepsilon^2/\log(1/\varepsilon))$ is chosen sufficiently small (in terms of the hidden constants) so that $|N_\delta|^{O(\varepsilon^{-2}\log(1/\varepsilon))} \leq (n^\rho \log n)^{O(\varepsilon^{-2}\log(1/\varepsilon))} < n-2$. By the pigeonhole principle, there is either a pair of points of P mapped to the same cell of U_B, under f, or there is a pair of points of P that are not mapped to U_B. In either case, the two points are not $(\varepsilon/2, 1+\varepsilon/20)$-stabbed by any point of $f(N_\delta)$. To see this, notice that if the two points are in the same cell, then they are within distance $\varepsilon/4$ from each other, contradicting the fact that they are $(\varepsilon/2, 1+\varepsilon/20)$-stabbed (which would mean they are separated by a distance of at least $\varepsilon/2$). If they are not mapped in U_B then neither of them is within distance $1+\varepsilon/20$ from any point of $f(N_\delta)$, also implying that they cannot be $(\varepsilon/2, 1+\varepsilon/20)$-stabbed by any point of $f(N_\delta)$. Since the two points are not $(\varepsilon/2, 1+\varepsilon/20)$-stabbed by any point of $f(N_\delta)$ in the mapped space, their pre-images (under f) are not ε-stabbed by any point of N_δ. Since N_δ is a δ-net, it must be that at most δm points of Q ε-stab those two points.

We now focus on the running time needed to find a pair of points as above. By Theorem 7, we can map points under f in $\text{poly}(n)$ time. To compute the two points, we essentially use the bucketing method (see, e.g., [20]) for balls centered at $f(N_\delta)$. We first store all cells of U_B in a dictionary data structure. Then, for each point $x \in f(P)$, we query the data structure to see if its cell C_x is stored, i.e., $C_x \in U_B$. If $C_x \in U_B$ then we store x in a linked list in the bucket of C_x. We can stop and output any two points stored in the same linked list, or any two points whose cells do not belong to U_B. Iterating over all points of $f(P)$ costs $\tilde{O}(dn)$ assuming that the dictionary is implemented using balanced binary trees. The overall probability of success follows from a union bound over failure events of the random sampling for computing N_δ and the failure event of computing the terminal embedding f. ◀

Knowing that a good pair exists, and that we can find it, we repeat the process $n/2$ times, in order to build a good forest, in the sense that it has a small ε-stabbing number. We use the MWU technique introduced in the context of trees with low stabbing number in [15].

▶ **Lemma 12.** *Let $P \subset \mathbb{R}^d$ be a set of n points and let $\varepsilon \in (0,1)$. There is a forest of trees on P with at least $n/2$ edges, such that every point of $\mathcal{G}_\gamma^d$ ε-stabs $\tilde{O}(dn^{1-\rho})$ edges, where $\gamma = \Theta(\varepsilon/\sqrt{d})$, $\rho = \Theta(\varepsilon^2/\log(1/\varepsilon))$. For any $c > 0$, we can compute such a forest in $\text{poly}(n, d, \frac{1}{\varepsilon})) + \tilde{O}(n) \cdot O(1/\varepsilon)^d$ time, with probability of success $1 - n^{-c}$.*

We conclude with the main result of this section, the tractability of a spanning tree with a sublinear ε-stabbing number. This follows from $O(\log n)$ iterative applications of Lemma 12.

▶ **Theorem 13.** *Let $P \subset \mathbb{R}^d$ be a set of n points. There is a spanning tree on P such that every point of $\mathcal{G}_\gamma^d$ ε-stabs $\tilde{O}(dn^{1-\rho})$ edges, where $\gamma = \Theta(\varepsilon/\sqrt{d})$, $\rho = \Theta(\varepsilon^2/\log(1/\varepsilon))$.*

Moreover, for any $c > 0$, we can compute such a spanning tree in $\text{poly}(n, d, \frac{1}{\varepsilon}) + \tilde{O}(n) \cdot O(1/\varepsilon)^d$ time, with probability of success $1 - n^{-c}$.

5 Spherical range counting

In this section, we show how to construct a partition tree for the approximate spherical range counting problem with sublinear query time. We first show how to get a partition tree with a sublinear visiting number given a spanning tree with a sublinear ε-stabbing number, by

extending the procedure in [15] to accommodate the approximation scheme. Subsequently, we incorporate the data structures of Section 3 in the inner nodes of the partition tree to get sublinear worst-case guarantees.

We start by proving that having a spanning tree with low ε-stabbing number can give us a spanning path with low ε-stabbing number, which we then use to construct a partition tree with low visiting number. Lemmas 14 and 15 are adapted from [15] for ε-stabbing numbers, and the bounds become dependent on the ambiguity zone of the query.

▶ **Lemma 14.** *Given a spanning tree T on a pointset $P \subset \mathbb{R}^d$, we can build a spanning path $\mathcal{P}$ on P, in $O(dn)$ time, such that for any point q, $\sigma(q, \mathcal{P}) \leq 2\sigma(q, T) + 2t_q$, where $t_q = |\,(B(q, 1+\varepsilon) \setminus B(q, 1)) \cap P|$.*

▶ **Lemma 15.** *Given a spanning path $\mathcal{P}$ on a pointset $P \subset \mathbb{R}^d$ we can build a binary partition tree $\mathcal{T}$ on P, in $(dn \log n)$ time, such that for any point q, we have $\zeta(q, \mathcal{T}) \leq 2[\sigma(q, \mathcal{P}) + t_q] \cdot \log n + 1$, where $t_q = |\,(B(q, 1+\varepsilon) \setminus B(q, 1)) \cap P|$.*

We combine these results to get a bound on the visiting number of a partition tree.

▶ **Corollary 16.** *Let $P \subset \mathbb{R}^d$ be a set of n points. For any $c > 0$, we can build a binary partition tree $\mathcal{T}$ on P, in $\mathrm{poly}(n, d, \frac{1}{\varepsilon}) + \tilde{O}(n) \cdot O(1/\varepsilon)^d$ time such that with probability $1 - 1/n^c$ every $q \in \mathcal{G}_\gamma^d$, with $\gamma = \varepsilon/\sqrt{d}$, visits $\zeta(q, \mathcal{T}) = \tilde{O}(dn^{1-\rho} + t_q)$ nodes, where $t_q = |\,(B(q, 1+\varepsilon) \setminus B(q, 1)) \cap P|$ and $\rho = \Theta(\varepsilon^2 / \log(1/\varepsilon))$.*

Finally, we present the main contribution of our paper, which states that for the approximate spherical range counting problem, we can build a data structure in polynomial time that supports sublinear queries (if t_q is also sublinear) and takes near-linear space. In our proof, we make use of Theorems 10 and 13 and Lemma 15.

▶ **Theorem 17.** *Let $P \subset \mathbb{R}^d$ be a set of n points, such that each $p \in P$ is assigned a weight $w_p \in \mathbb{R}$, and let $\varepsilon \in (0, 1)$. We can build a data structure for the approximate range counting problem with input P, $\{w_p \mid p \in P\}$, ε, that uses space in $\tilde{O}(n)$, needs preprocessing time in $O(dn) + n^{O(\varepsilon^{-2} \log(1/\varepsilon))}$ and for each query $q \in \mathbb{R}^d$, it returns an answer in $n^{1-\Theta(\varepsilon^4/\log(1/\varepsilon))} + t_q^{\varrho} \cdot n^{1-\varrho}$ time, where $\varrho = \Theta(\varepsilon^2)$ and $t_q = |\,(B(q, 1+\varepsilon) \setminus B(q, 1)) \cap P|$. For each query, the preprocessing succeeds with probability $1 - n^{-c}$, where $c > 0$ is a constant hidden in the aforementioned time and space complexities.*

Proof. We first apply a JL transformation [28] to reduce the dimension to $O(\varepsilon^{-2} \log n)$. By Theorem 6, and by applying a union bound over n vectors, for any $c > 0$ and any query $q \in \mathbb{R}^d$, with probability $1 - 1/n^c$, all distances from P are approximately preserved within factors $1 \pm \varepsilon/10$. This initial step costs $O(dn)$ time. For the rest of the proof, we assume that all points lie in $\mathbb{R}^{O(\varepsilon^{-2} \log n)}$. Then, we assume that queries are snapped to the grid $\mathcal{G}_\gamma^d$, $\gamma = \varepsilon/(10\sqrt{d})$. This induces an additive $\pm\varepsilon/10$ error to the distances between P and the query. Hence, points within distance 1 from the query may be moved to a distance up to $1 + \varepsilon/5$, and points at a distance of at least $1 + \varepsilon$ may be moved to a distance of at least $1 + 4\varepsilon/5$. Notice that $1 + \varepsilon/2 \leq \frac{1+4\varepsilon/5}{1+\varepsilon/5}$, hence it suffices to use a more refined error of $\varepsilon/2$ for the rest of the construction and assume that the space is rescaled by a factor $1/(1 + \varepsilon/5)$.

We build a binary partition tree $\mathcal{T}$ on the input with error parameter $\varepsilon/2$ using Corollary 16. The construction of $\mathcal{T}$ can be done in $n^{O(\varepsilon^{-2} \log(1/\varepsilon))}$ time and be successful for a fixed q, in that $\zeta(q, \mathcal{T}) \leq n^{1-\Theta(\varepsilon^2/\log(1/\varepsilon))} + t_q$, with probability $1 - 1/n^c$ for any constant $c > 0$. For each internal node v, let P_v be the set of points in the leaves of the subtree rooted at v. In each internal node v, we build a data structure $\mathcal{D}_v$ as in Theorem 10 on P_v with error

parameter $\varepsilon/2$, and we store the cumulative weight s_v of the points in P_v, i.e., $s_v = \sum_{p \in P_v} w_p$. By Theorem 10, each construction of a data structure $\mathcal{D}_v$, can be done in $\tilde{O}(d|P_v|)$ time and be successful with probability $1 - 1/n^c$ for any constant $c > 0$. By a union bound over all inner nodes, we claim that all data structures $\mathcal{D}_v$ are also successful for q with probability $1 - \Theta\left(\frac{n \log n}{n^c}\right)$. By Theorem 10, the time and space needed to build data structures for the inner nodes is near-linear per level of the tree (since each level defines a partition).

To answer a query $q \in \mathbb{R}^d$, we traverse the tree and collect all cumulative weights s_v in each internal node v, whose associated data structure $\mathcal{D}_v$ returns "covered". We continue traversing the children of a node v if and only if $\mathcal{D}_v$ returns "stabbed". By Corollary 16, assuming that the tree has been constructed correctly for $q \in \mathbb{R}^d$, the number of nodes visited during the query algorithm for input $q \in \mathbb{R}^d$ is $\tilde{O}(n^\alpha + t_q)$, where $\alpha = 1 - \Theta(\varepsilon^2/\log(1/\varepsilon))$. By Theorem 10, the time needed to query a data structure $\mathcal{D}_v$ in an internal node v is $\tilde{O}(|P_v|^\beta)$, where $\beta = 1 - \Theta(\varepsilon^2)$. At each level i of $\mathcal{T}$, the number of nodes whose data structures are queried is upper bounded by both 2^i and $\zeta(q, \mathcal{T})$. Hence, the running time at level i for query q is $\tilde{O}(T_i^q(n))$, where $T_i^q(n)$ is defined as follows:

$$T_i^q(n) = \min\left\{2^i,\ n^\alpha + t_q\right\} \cdot \left(\frac{n}{2^i}\right)^\beta \leq (n^\alpha + t_q)^{1-\beta} \cdot n^\beta,$$

because if $i \leq \log(n^\alpha + t_q)$ then $\min\left\{2^i,\ n^\alpha + t_q\right\} = 2^i$ and if $i > \log(n^\alpha + t_q)$ then $\min\left\{2^i,\ n^\alpha + t_q\right\} = n^\alpha + t_q$. Now, since $1 - \beta < 1$,

$$T_i^q(n) \leq n^{\alpha(1-\beta)+\beta} + t_q^{1-\beta} \cdot n^\beta = n^{1-\Theta(\varepsilon^4/\log(1/\varepsilon))} + t_q^\varrho \cdot n^{1-\varrho},$$

where $\varrho = 1 - \beta$. We observe that the query algorithm will not explore any descendants of an internal node v if $\mathcal{D}_v$ returns "disjoint" when $\min_{x \in P_v} \|x - q\| > 1$, or if $\mathcal{D}_v$ returns "covered" when $\max_{x \in P_v} \|x - q\| \leq 1 + \varepsilon/2$, in which case it takes into account s_v. Thus, the answer is calculated by summing over cumulative weights $s_{v_1}, \ldots, s_{v_\ell}$ of nodes corresponding to subsets $P_{v_1}, \ldots, P_{v_\ell}$ such that $B(q, 1) \cap P \subseteq \bigcup_{i \in [\ell]} P_{v_i} \subseteq B(q, 1 + \varepsilon/2) \cap P$. ◀

6 Data-driven range searching

As we saw in the previous section, we can construct the partition tree in polynomial time. However, the exponent depends on $1/\varepsilon$, which we would like to avoid. In this section, we present an algorithm inspired by learning theory that constructs a partition tree with near-optimal visiting number in expectation, over the distribution of queries. Specifically, the algorithm will have access to a number m of sample queries, and will construct a partition tree that is optimal for that sample - thus the algorithm is data-driven. Showing that uniform convergence is achieved, we prove that this partition tree is nearly optimal, in that its expected visiting number is nearly minimal.

In the next lemma, we show that using only a near-linear number of queries we can achieve uniform convergence for the expected ε-stabbing number, i.e., for any spanning tree, the estimation of the ε-stabbing number using only these queries is close to its real expected ε-stabbing number.

▶ **Lemma 18.** *Let P be a set of n points in $\mathbb{R}^d$ and let $\mathcal{D}_Q$ be a ball query distribution having as support a countably infinite set $\mathcal{R}$. It suffices to sample $m \in O(n \cdot (d \log n + \log(1/\delta)))$ queries S independently from $\mathcal{D}_Q$, so that with probability at least $1 - \delta$, for any spanning tree T of P, $m^{-1} \cdot \sum_{q \in S} \sigma(q, T)$ is in $\left[\frac{5}{8} \cdot \mathbb{E}_{q \sim \mathcal{D}_Q}[\sigma(q, T)] - \frac{3}{8}, \frac{11}{8} \cdot \mathbb{E}_{q \sim \mathcal{D}_Q}[\sigma(q, T)] + \frac{3}{8}\right]$.*

The next lemma states that the number of nodes of a partition tree visited during a query search consists an upper bound on the ε-stabbing number of a corresponding spanning path of the points for the same query. Its proof is a straightforward adaptation of the third item of [15, Lemma 3.1].

▶ **Lemma 19.** *If $\mathcal{T}$ is a partition tree for P, then there exists a spanning path $\mathcal{P}$ whose ε-stabbing number does not exceed the visiting number of $\mathcal{T}$.*

We now prove the main result of this section.

▶ **Theorem 20.** *Let P be a set of n points in $\mathbb{R}^d$ and let $\mathcal{D}_Q$ be a ball query distribution having as support a countably infinite set $\mathcal{R}$. It suffices to sample a set S of $m \in O(n \cdot (d \log n + \log(1/\delta)))$ queries independently from $\mathcal{D}_Q$, so that with probability at least $1-\delta$, we can compute a partition tree $\tilde{\mathcal{T}}$ such that*

$$\mathbb{E}_{q\sim\mathcal{D}_Q}\left[\zeta(q,\tilde{\mathcal{T}})\right] = O\left(\left[\min_{\mathcal{T}} \mathbb{E}_{q\sim\mathcal{D}_Q}[\zeta(q,\mathcal{T})] + t_q\right] \cdot \log n\right),$$

where $t_q = |\,(B(q,1+\varepsilon) \setminus B(q,1)) \cap P|$. Moreover, given S, the running time to compute $\tilde{\mathcal{T}}$ is in $O(n^3(d\log n + \log(1/\delta))$.

Proof. By Lemma 18, with probability at least $1-\delta$, for any spanning tree T of P, $m^{-1} \cdot \sum_{q\in S} \sigma(q,T)$ is in

$$\left[\frac{5}{8} \cdot \mathbb{E}_{q\sim\mathcal{D}_Q}\left[\sigma(q,T)\right] - \frac{3}{8}, \frac{11}{8} \cdot \mathbb{E}_{q\sim\mathcal{D}_Q}\left[\sigma(q,T)\right] + \frac{3}{8}\right].$$

Now, let $E(T)$ denote the edges of spanning tree T and $\tilde{T} = \arg\min_T\{m^{-1} \cdot \sum_{q\in S} \sigma(q,T)\}$. Notice that $\sum_{q\in S} \sigma(q,T) = \sum_{q\in S} \sum_{e\in E(T)} \sigma(q,\{e\}) = \sum_{e\in E(T)} \sum_{q\in S} \sigma(q,\{e\})$ for any spanning tree T. Hence, we can compute $\tilde{T}$, by first constructing a complete weighted graph on P where we assign weight $\sum_{q\in S} \sigma(q,\{e\})$ to each edge $e \in \binom{P}{2}$, and then computing the minimum spanning tree on that graph, using Kruskal's algorithm. The running time of this step is $O(n^2 m)$, which corresponds to the step of building the graph.

Let T^* be the spanning tree that minimizes $\mathbb{E}_{q\sim\mathcal{D}_Q}[\sigma(q,T)]$ over all spanning trees T of P. By Lemma 19 we get $\mathbb{E}_{q\sim\mathcal{D}_Q}[\sigma(q,T^*)] \le \min_{\mathcal{T}} \mathbb{E}_{q\sim\mathcal{D}_Q}[\zeta(q,\mathcal{T})]$ and by Lemma 18 we get:

$$\Omega(1) \cdot \mathbb{E}_{q\sim\mathcal{D}_Q}[\sigma(q,\tilde{T})] \le m^{-1} \sum_{q\in S} \sigma(q,\tilde{T}) \le m^{-1} \sum_{q\in S} \sigma(q,T^*) \le O(1) \cdot \mathbb{E}_{q\sim\mathcal{D}_Q}[\sigma(q,T^*)]$$

Hence $\mathbb{E}_{q\sim\mathcal{D}_Q}[\sigma(q,\tilde{T})] \subseteq O\left(\min_{\mathcal{T}} \mathbb{E}_{q\sim\mathcal{D}_Q}[\zeta(q,\mathcal{T})]\right)$. Given $\tilde{T}$, by Lemmas 14 and 15 we can construct a binary partition tree $\tilde{\mathcal{T}}$ of P such that

$$\begin{aligned}\mathbb{E}_{q\sim\mathcal{D}_Q}[\zeta(q,\tilde{\mathcal{T}})] &= O\left(\left[\mathbb{E}_{q\sim\mathcal{D}_Q}[\sigma(q,\tilde{T})] + t_q\right] \cdot \log n\right) \\ &= O\left(\left[\min_{\mathcal{T}} \mathbb{E}_{q\sim\mathcal{D}_Q}[\zeta(q,\mathcal{T})] + t_q\right] \cdot \log n\right)\end{aligned}$$

in $O(n)$ time. Thus, the overall running time is $O(n^2 m) \subseteq O(n^3(d\log n + \log(1/\delta))$. ◀

References

1 Peyman Afshani, Jérémy Barbay, and Timothy M. Chan. Instance-optimal geometric algorithms. *J. ACM*, 64(1), 2017. doi:10.1145/3046673.

2 Thomas D. Ahle, Martin Aumüller, and Rasmus Pagh. Parameter-free locality sensitive hashing for spherical range reporting. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 239–256. SIAM, 2017. doi:10.1137/1.9781611974782.16.

3 Nir Ailon, Bernard Chazelle, Kenneth L. Clarkson, Ding Liu, Wolfgang Mulzer, and C. Seshadhri. Self-improving algorithms. *SIAM J. Comput.*, 40(2):350–375, 2011. doi:10.1137/090766437.

4 Tomas Akenine-Möller, Eric Haines, Naty Hoffman, Angelo Pesce, Michał Iwanicki, and Sébastien Hillaire. *Real-Time Rendering 4th Edition*. A K Peters/CRC Press, Boca Raton, FL, USA, 2018.

5 J. Alman, T. M.Chan, and R. R. Williams. Polynomial representations of threshold functions and algorithmic applications. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 467–476, 2016. doi:10.1109/FOCS.2016.57.

6 A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008. doi:10.1145/1327452.1327494.

7 Alexandr Andoni and Daniel Beaglehole. Learning to hash robustly, guaranteed. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 599–618. PMLR, 2022. URL: https://proceedings.mlr.press/v162/andoni22a.html.

8 Alexandr Andoni, Thijs Laarhoven, Ilya P. Razenshteyn, and Erik Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 47–66. SIAM, 2017. doi:10.1137/1.9781611974782.4.

9 Martin Anthony and Peter L. Bartlett. *Neural Network Learning - Theoretical Foundations*. Cambridge University Press, 2002. URL: http://www.cambridge.org/gb/knowledge/isbn/item1154061/?site_locale=en_GB.

10 Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM Journal on Computing*, 38(3):899–921, 2008. doi:10.1137/060669474.

11 Sunil Arya and David M. Mount. Approximate range searching. *Comput. Geom.*, 17(3-4):135–152, 2000. doi:10.1016/S0925-7721(00)00022-5.

12 Maria-Florina Balcan, Dan F. DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. How much data is sufficient to learn high-performing algorithms? *J. ACM*, 71(5):32:1–32:58, 2024. doi:10.1145/3676278.

13 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.

14 Timothy M Chan. Optimal partition trees. In *Proceedings of the twenty-sixth annual symposium on Computational geometry*, pages 1–10, 2010. doi:10.1145/1810959.1810961.

15 Bernard Chazelle and Emo Welzl. Quasi-optimal range searching in space of finite VC-dimension. *Discret. Comput. Geom.*, 4:467–489, 1989. doi:10.1007/BF02187743.

16 Qi Cheng. On comparing sums of square roots of small integers. In *Proceedings of the 31st International Conference on Mathematical Foundations of Computer Science*, MFCS'06, pages 250–255, Berlin, Heidelberg, 2006. Springer-Verlag. doi:10.1007/11821069_22.

17 Mónika Csikós and Nabil H. Mustafa. Escaping the curse of spatial partitioning: Matchings with low crossing numbers and their applications. In Kevin Buchin and Éric Colin de Verdière, editors, *37th International Symposium on Computational Geometry, SoCG 2021, June 7-11, 2021, Buffalo, NY, USA (Virtual Conference)*, volume 189 of *LIPIcs*, pages 28:1–28:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.SOCG.2021.28.

18 Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In Jack Snoeyink and Jean-Daniel Boissonnat, editors, *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*, pages 253–262. ACM, 2004. doi:10.1145/997817.997857.

19 Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM J. Comput.*, 46(3):992–1017, 2017. doi:10.1137/15M1050276.

20 S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012. doi:10.4086/toc.2012.v008a014.

21 Sariel Har-Peled. Approximating spanning trees with low crossing number, 2009. arXiv:0907.1131.

22 Sariel Har-peled. *Geometric Approximation Algorithms*. American Mathematical Society, USA, 2011.

23 Sariel Har-Peled and Micha Sharir. Relative (p,ϵ)-Approximations in Geometry. *Discrete & Computational Geometry*, 45(3):462–496, April 2011. doi:10.1007/S00454-010-9248-1.

24 David Haussler and Emo Welzl. ε-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987. URL: https://api.semanticscholar.org/CorpusID:27638326.

25 O. Huisman and R.A. de By. *Principles of geographic information systems : an introductory textbook*. ITC Educational Textbook Series. International Institute for Geo-Information Science and Earth Observation, Netherlands, 2009.

26 P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annual ACM Symp. Theory of Computing*, STOC'98, pages 604–613, 1998. doi:10.1145/276698.276876.

27 P. Indyk and A. Naor. Nearest-neighbor-preserving embeddings. *ACM Trans. Algorithms*, 3(3), 2007. doi:10.1145/1273340.1273347.

28 William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into hilbert space. *Contemporary mathematics*, 26:189–206, 1984. URL: https://api.semanticscholar.org/CorpusID:117819162.

29 Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 309–318, USA, 2000. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=338219.338267.

30 Honghao Lin, Tian Luo, and David P. Woodruff. Learning augmented binary search trees. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 13431–13440. PMLR, 2022. URL: https://proceedings.mlr.press/v162/lin22f.html.

31 Qiyu Liu, Maocheng Li, Yuxiang Zeng, Yanyan Shen, and Lei Chen. How good are multi-dimensional learned indexes? an experimental survey. *VLDB J.*, 34(2):17, 2025. doi:10.1007/S00778-024-00893-6.

32 Jirí Matousek. Efficient partition trees. *Discret. Comput. Geom.*, 8:315–334, 1992. doi:10.1007/BF02293051.

33 Shyam Narayanan and Jelani Nelson. Optimal terminal dimensionality reduction in euclidean space. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1064–1069. ACM, 2019. doi:10.1145/3313276.3316307.

34 Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory Series A*, 13:145–147, 1972. doi:10.1016/0097-3165(72)90019-2.

35 Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1), 1972.

36 Abraham Silberschatz, Henry Korth, and S. Sudarshan. *Database Systems Concepts*. McGraw-Hill, Inc., USA, 5 edition, 2005.

37 Vladimir Vapnik and Alexey Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

38 Emo Welzl. Partition trees for triangle counting and other range searching problems. In Herbert Edelsbrunner, editor, *Proceedings of the Fourth Annual Symposium on Computational Geometry, Urbana-Champaign, IL, USA, June 6-8, 1988*, pages 23–33. ACM, 1988. doi:10.1145/73393.73397.

39 Emo Welzl. On spanning trees with low crossing numbers. In Burkhard Monien and Thomas Ottmann, editors, *Data Structures and Efficient Algorithms, Final Report on the DFG Special Joint Initiative*, volume 594 of *Lecture Notes in Computer Science*, pages 233–249. Springer, 1992. doi:10.1007/3-540-55488-2_30.

Complements of Finite Unions of Convex Sets

Chaya Keller
School of Computer Science, Ariel University, Israel

Micha A. Perles
Einstein Institute of Mathematics, Hebrew University, Jerusalem, Israel

Abstract

Finite unions of convex sets are a central object of study in discrete and computational geometry. In this paper we initiate a systematic study of *complements* of such unions – i.e., sets of the form $S = \mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$, where K_i are convex sets. In the first part of the paper we study isolated points in S, whose number is related to the Betti numbers of $\cup_{i=1}^n K_i$ and to its non-convexity properties. We obtain upper bounds on the number of such points, which are sharp for $n = 3$ and significantly improve previous bounds of Lawrence and Morris (2009) for all $n \ll \frac{2^d}{d}$. In the second part of the paper we study coverings of S by well-behaved sets. We show that S can be covered by at most $g(d, n)$ flats of different dimensions, in such a way that each $x \in S$ is covered by a flat whose dimension equals the "local dimension" of S in the neighborhood of x. Furthermore, we determine the structure of a minimum cover that satisfies this property. Then, we study quantitative aspects of this minimum cover and obtain sharp upper bounds on its size in various settings.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases convexity, unions of convex sets

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.61

Related Version *Full Version*: https://arxiv.org/pdf/2508.19413 [10]

Funding *Chaya Keller*: Partially supported by Grant 1065/20 from the Israel Science Foundation.

1 Introduction

The complexity of finite unions of convex sets has been a prolific research area in the last decades, due to intrinsic deep discrete-geometric questions pertaining to it and to applications to optimization, robotics, motion planning and other areas (see, e.g., the survey [1]). In some of these questions and applications, the structure of the complement plays an important role, and as a result, various structural questions regarding complements of such finite unions were studied in different works over the years.

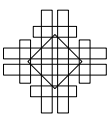
One well-studied question is determining the maximum possible number of *bounded connected components* in $S = \mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$ (also known as *voids* in $K = \cup_{i=1}^n K_i$), where $\{K_i\}$ are convex. Asked by Fejes Tóth in the plane, and independently by Vitushkin (1958) in $\mathbb{R}^d$, this question was studied both in full generality and for specific families of convex sets, such as translates of the same convex set (see, e.g., [2, 9]). As the number of voids is equal to the $(d-1)$'st Betti number of $\cup_{i=1}^n K_i$, bounding its maximum size is the first step toward understanding the statistical behavior of the Betti numbers of such unions (see [8]).

A related question is determining the maximum possible number of *all* connected components (including unbounded ones). Kovalev [11] proved that the maximum is $\sum_{i=0}^{d} \binom{n}{i}$, and that it is attained if and only if K_i are either hyperplanes in general position, or layers between parallel hyperplanes. Bei, Chen and Zhang [5] provided a different proof, and applied the result to analyze the complexity of an algorithm they proposed for solving linear programming problems with only partial knowledge of the constraints.

In the case where $\{K_i\}$ are polyhedra, the combinatorial complexity of S plays an important role in motion planning, and was studied, e.g., by Aronov and Sharir in [3, 4].



© Chaya Keller and Micha A. Perles;
licensed under Creative Commons License CC-BY 4.0



The case where S is finite was studied by Lawrence and Morris [12], who obtained upper bounds on $|S|$ (which, in this case, is equal to the number of *one-point holes* in K) in terms of n. As was shown much earlier by Matoušek and Valtr [13, Thm. 1.1(ii)], bounds on the number of one-point holes allow bounding the *convexity number* of K in terms of its *invisibility number* (see also [7, Theorem 1]).

In this paper we initiate a systematic study of the structural properties of such sets S. We concentrate on two types of questions – isolated points in S (which were studied in the special case where S is finite by Lawrence and Morris [12]), and covering of S by flats (i.e., affine subspaces of $\mathbb{R}^d$).

Isolated points in S. For the sake of convenience, we use the following definition:

▶ **Definition 1.** *A point $p \in \mathbb{R}^d$ is* encapsulated *by the convex sets $K_1, \ldots, K_n \subset \mathbb{R}^d$ if for some $\epsilon > 0$, $B(p,\epsilon) \setminus (\cup_{i=1}^n K_i) = \{p\}$, where $B(p,\epsilon)$ is the open ball of radius ϵ centered at p. In other words, $p \in \mathbb{R}^d$ is* encapsulated *by $K_1, \ldots, K_n \subset \mathbb{R}^d$ if these convex sets cover a pointed neighborhood of p.*

Lawrence and Morris [12] studied the case $|S| < \infty$, in which every $p \in S$ is encapsulated by the convex sets $K_1, \ldots, K_n$. In the case where all $\{K_i\}$ are open, they used a classical theorem of Björner and Kalai [6] to show that $|S| \leq \binom{n-1}{d}$. On the other hand they showed the existence of such a set S with $(\lfloor \frac{n}{d} \rfloor - 1)^d \leq |S|$. With no additional assumption on $\{K_i\}$ (except for convexity), but assuming that S affinely spans $\mathbb{R}^d$, they obtained the upper bound $|S| \leq (n-1)(\binom{n}{\lfloor \frac{n}{2} \rfloor})^{d-1}$.

We consider the general case, where no additional assumptions on S and $\{K_i\}$ are made. First, we obtain the following tight bound on the number of points encapsulated by 3 convex sets:

▶ **Theorem 2.** *Let $K_1, K_2, K_3 \subset \mathbb{R}^d$ be 3 convex sets. Denote by S the set of points of $\mathbb{R}^d$ encapsulated by $K_1 \cup K_2 \cup K_3$. Define $f(d) = \lfloor \frac{3d}{2} \rfloor + 1$. Then:*

(a) $|S| \leq f(d)$;

(b) If $|S| = f(d)$ then the sets K_1, K_2, K_3 are pairwise disjoint;

(c) The sets K_1, K_2, K_3 can be chosen in such a way that $|S| = f(d)$ and $\mathbb{R}^d \setminus (K_1 \cup K_2 \cup K_3) = S$.

For larger numbers of convex sets, let $f(d,n)$ be the maximal number of points that can be encapsulated by n convex sets in $\mathbb{R}^d$. It is easy to see[1] that for $d \geq 1$, $f(d,1) = 0$, $f(d,2) = 1$, for all $n \geq 1$ we have $f(1,n) = n-1$ and by Theorem 2, $f(d,3) = \lfloor \frac{3d}{2} \rfloor + 1$. The following recursive bound can be proved inductively, by determining arbitrarily $f(0,n) = 1$.

▶ **Theorem 3.** *For every $n > 3, d \geq 2$,*

$$f(d,n) \leq \sum_{i=2}^{n-1} \left(\binom{n}{i} f(d,i) \right) + f(d-2,n),$$

and consequently,

$$f(d,n) \leq 2d^{n-2}n!.$$

[1] See also Theorem 9 below.

Note that for a constant n, the bound we obtain is polynomial in d, whereas the bound of [12] is exponential in d. Furthermore, a direct calculation shows that our bound is superior as long as $n \ll \frac{2^d}{d}$.

In addition, we prove a sharp bound in the plane, under the additional assumptions that S is finite and the sets K_i are pairwise disjoint. It turns out that while in $\mathbb{R}^1$ and for three sets in $\mathbb{R}^2$, the maximal size of S is obtained where the convex sets are pairwise disjoint, in the general case the disjointness assumption leads to a much smaller bound on $|S|$ – even in the plane, where the upper bound is linear in n (compared to a quadratic lower bound without this restriction, see [10, Appendix A]).

▶ **Proposition 4.** *Let $S = \mathbb{R}^2 \setminus (\cup_{i=1}^n K_i)$ where $n \geq 3$ and $\{K_i\}$ are pairwise disjoint convex sets. Assume that $|S| < \infty$. Then $|S| \leq 5n - 11$, and this bound is sharp.*

Covering S by flats. For the sake of convenience, we use the following definition.

▶ **Definition 5.** *For $S \subset \mathbb{R}^d$, the* flat-dimension *of S, $dm(S)$, is*

$$\mathrm{dm}(S) = max\{k : S \text{ includes a } k\text{-simplex}\},$$

where a k-simplex (for $k \geq -1$) is the convex hull of $k+1$ affinely independent points in $\mathbb{R}^d$.

The local dimension $\mathrm{dm}(S, p)$ *of S at $p \in S$ is*

$$\mathrm{dm}(S, p) = \min_{\epsilon > 0} \mathrm{dm}(S \cap B(p, \epsilon)).$$

Our main result in this part is the following theorem:

▶ **Theorem 6.** *For any $n, d \in \mathbb{N}$, there exists a number $g = g(n, d)$ such that the following holds. Let $S = \mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$, where $\{K_i\}$ are convex sets. Then S can be covered by at most g flats, in such a way that each $p \in S$ is covered by a flat of dimension $\mathrm{dm}(S, p)$. Furthermore, there exists a unique such cover $\mathcal{C}$ that is minimal with respect to inclusion.*

Actually, the proof of Theorem 6 provides significant structural information on $\mathcal{C}$: Suppose $p \in S, \mathrm{dm}(S, p) = k$, and let L be a flat. If $S \cap W = L \cap W$ for some neighborhood W of p, then L is the k-flat in $\mathcal{C}$ that covers p.

On the quantitative side, the bound on $g(d, n)$ which follows from the proof is rather large, and in particular, the maximum numbers of flats of each dimension in $\mathcal{C}$ seem difficult to compute in general. Hence, we focus on special cases where effective bounds can be obtained. The following natural definition will be convenient.

▶ **Definition 7.** *For $S = \mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$, where $\{K_i\}$ are convex sets, and for $k = 0, 1, \ldots, d$, denote by $\nu_k(S)$ the number of k-flats in the cover $\mathcal{C}$ of S. Denote by $\nu_k(d, n)$ the maximum of $\nu_k(S)$ over all such sets S, where d, n, k are fixed.*

Theorems 2 and 3 above provide upper bounds on $\nu_0(d, n)$ (i.e., on the maximal number of 0-dimensional flats in $\mathcal{C}$), as by definition, the flat which covers each isolated point in S must be the point itself. At the other end of the spectrum, it is clear that $\nu_d(d, n) = 1$. Regarding $(d-1)$-flats, we determine $\nu_{d-1}(d, n)$ completely.

▶ **Theorem 8.** *We have*

$$\nu_1(2, n) = t(n, 4) = (\tfrac{3}{4} + o(1))\binom{n}{2}, \quad \text{and} \quad \nu_{d-1}(d, n) = \binom{n}{2}, \quad \forall d \geq 3,$$

where $t(n, 4)$ is the Turán number.

Finally, in the case $n = 2$, i.e., where there are only two convex sets, we completely determine the numbers of k-flats that can appear in the cover $\mathcal{C}$ of $S = \mathbb{R}^d \setminus (K_1 \cup K_2)$, for each $0 \leq k \leq d$.

▶ **Theorem 9.** *Let K_1, K_2 be convex sets in $\mathbb{R}^d$ ($d \geq 1$), and let $\mathcal{C}$ be the cover of $S = \mathbb{R}^d \setminus (K_1 \cup K_2)$ discussed above. Then any two flats $I, J \in \mathcal{C}$ satisfy $J \subset I$ or $I \subset J$. Consequently, $\mathcal{C}$ contains at most one k-flat for each $0 \leq k \leq d$.*

Moreover, for any subset T of $\{0, 1, \ldots, d\}$ we can find $K_1, K_2 \subset \mathbb{R}^d$ such that $\mathcal{C}$ contains a k-flat if and only if $k \in T$.

The rest of the paper is organized as follows. In Section 2 we study isolated points in S, proving Theorems 2 and 3. In Section 3 we study the qualitative question of covering S by flats, proving Theorem 6. In Sections 4 and 5 we study quantitative aspects of covering S by flats and present the proofs of Theorems 8 and 9.

2 Bounding the number of points encapsulated by convex sets

Recall that $f(d, n)$ is the maximal number of points that can be encapsulated by n convex sets in $\mathbb{R}^d$. It is easy to see that for every $n > 0$, $f(1, n) = n - 1$ and that for every $d > 0$, $f(d, 2) = 1$. After introducing a few definitions and an observation in Section 2.1, we determine $f(d, 3)$ exactly in Section 2.2. Then, in Section 2.3 we obtain by an inductive argument an upper bound on $f(d, n)$ for $n > 3$.

2.1 Preliminaries

For a set $S \subset \mathbb{R}^d$, denote by $\mathrm{cl}(S)$ and $\mathrm{int}(S)$ the topological closure and interior of S, respectively. The convex hull of S is denoted by $\mathrm{conv}(S)$. For $d \in \mathbb{N}$ and $0 \leq k \leq d$, a k-flat $\pi \subset \mathbb{R}^d$ is a k-dimensional affine subspace of $\mathbb{R}^d$.

▶ **Definition 10.** *For $K \subset \mathbb{R}^d, p \in \mathbb{R}^d$, we say that p touches K (or K touches p), if $p \in cl(K) \setminus K$.*

The following observation will be used several times in the sequel.

▶ **Observation 11.** *Let $K_1, \ldots, K_n \subset \mathbb{R}^d$ be convex sets, and let B be a d-dimensional ball $B \subset K_1 \cap \ldots \cap K_n$. Let $p_0 \notin K_1 \cup \ldots \cup K_n$. Consider the double cone with apex p_0 spanned by B. Then the other side of this cone (the dark area in Figure 1) is disjoint to $K_1, \ldots, K_n$.*

Indeed, for each x in the other side of the cone above, p_0 lies inside a segment that connects x to a point in each K_i.

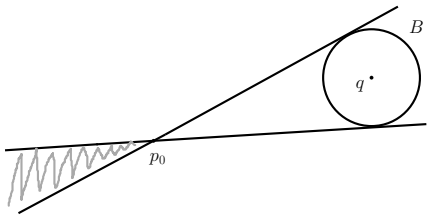


Figure 1 A cone with a vertex in p_0 that is tangent to a ball B.

2.2 The number of points encapsulated by 3 convex sets in $\mathbb{R}^d$

The following theorem implies that $f(d,3) = \lfloor \frac{3d}{2} \rfloor + 1$. For the sake of simplicity, we denote by S the set of encapsulated points, instead of the entire set $\mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$. The reason for abusing notation is that in the proof of this theorem, elements of $\mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$ other than encapsulated points do not make any difference. Hence, we implicitly assume that there are no such elements, and thus, $\mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$ is equal to the set of encapsulated points.

Proof of Theorem 2. We prove the first two statements together by induction on d. The cases $d = 0, 1$ are trivial. For $d \geq 2$ and for every $p \in S$, let

$$touch(p) = \{i : 1 \leq i \leq 3, \text{ and } p \textit{ touches } K_i\}.$$

We shall use the following observations.

▶ **Observation 12.** *If p is encapsulated by K_1, K_2, K_3 then $|touch(p)| \geq 2$, and if $touch(p) = \{i, j\}$ then $K_i \cap K_j = \emptyset$.*

Indeed, if $q \in K_i \cap K_j$ then all the points on the line $\ell(p, q)$ that are separated from q by p, are not in $K_i \cup K_j$. Since there exist such points arbitrarily close to p, this contradicts the assumption on p.

▶ **Observation 13.** *If for $p \in S$, $touch(p) = \{i, j\}$ then each of the cones $cone(p, K_i) = \{(1-\lambda)p + \lambda x : \lambda > 0, x \in K_i\}, cone(p, K_j)$ is a semi-space[2]. Moreover, there is no other point $p' \in S$ with $touch(p') = \{i, j\}$. (See Figure 2.)*

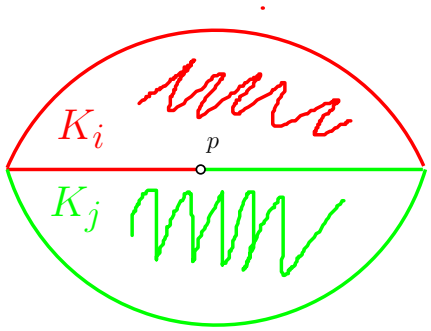


Figure 2 An illustration for Observation 13 in $\mathbb{R}^2$ where $touch(p) = \{i, j\}$. The set K_i is colored with red and K_j is colored with green. In this case $cone(K_i)$ is the upper half-plane, and $cone(K_j)$ is the lower half-plane.

By Observation 13, there is at most one point $p_{12} \in S$ with $touch(p_{12}) = \{1, 2\}$, at most one point $p_{13} \in S$ with $touch(p_{13}) = \{1, 3\}$, and at most one point $p_{23} \in S$ with $touch(p_{23}) = \{2, 3\}$. By Observation 12, all other points in S touch all three convex sets K_1, K_2, K_3. Let $S' = \{p \in S : |touch(p)| = 3\}$ and let $J = \text{aff}(S')$ be the flat spanned by S'. If $S' = J = \emptyset$ then we are done. Otherwise, by Observation 11, J cannot contain a d-dimensional ball, and hence, $\dim(J) < d$.

[2] In the plane, this means that each of $cone(p, K_i), cone(p, K_j)$ is a half-open half-space. In general dimension, this means that there exists a unique orthonormal base $B = \{e_1, \ldots, e_d\}$ of $\mathbb{R}^d$ such that with respect to B, $cone(p, K_i)$ is the set of all points that are lexicographically smaller than p, and $cone(p, K_j)$ is the set of all points that are lexicographically larger than p.

Consider $S \cap J$. This set includes S' and maybe some of the points p_{ij} defined above. Every point in $S \cap J$ is encapsulated (w.r.t. J) by the three convex sets $K_i \cap J$ $(1 \leq i \leq 3)$. Hence, by the induction hypothesis, $|S \cap J| \leq f(\dim(J))$. There are 3 cases:

Case 1: $\dim(J) = d - 2$. Then $|S'| \leq f(d-2)$, and since S contains at most three points that are not in S' (i.e., the p_{ij}'s above), we have

$$|S| \leq |S'| + 3 \leq f(d-2) + 3 = f(d).$$

If all inequalities hold with equality, then all three points p_{12}, p_{13} and p_{23} exist, and by Observation 12, the K_i's are pairwise disjoint.

Case 2: $\dim(J) < d - 2$. Then

$$|S| \leq |S'| + 3 \leq f(\dim J) + 3 < f(d).$$

Case 3: $\dim(J) = d - 1$. In this case we prove that the points p_{12}, p_{13} and p_{23} (if exist) are in J, and therefore, $|S| \leq f(d-1) < f(d)$, and we are done.

Indeed, assume to the contrary (w.l.o.g.) that p_{12} exists and $p_{12} \notin J$. Then $\dim(\mathrm{conv}(S' \cup \{p_{12}\})) = d$. Moreover, since all the points in S' touch both K_1 and K_2, we have

$$\mathrm{conv}(S' \cup \{p_{12}\}) \subset \mathrm{cl}(K_1) \cap \mathrm{cl}(K_2).$$

Therefore, $\dim(\mathrm{cl}(K_1) \cap \mathrm{cl}(K_2)) = d$, hence $K_1 \cap K_2 \neq \emptyset$, in contradiction to Observation 12. This completes the proof of parts (a) and (b) of the theorem.

For the proof of (c) we need the following claim:

▷ Claim 14. Let $A, B \subset \mathbb{R}^d$ be convex sets such that $A \cap B = \emptyset$, and let $z \in \mathbb{R}^d \setminus (A \cup B)$. Then there exist two convex sets $\tilde{A}, \tilde{B} \subset \mathbb{R}^d$ such that

- $A \subset \tilde{A}, B \subset \tilde{B}$,
- $\tilde{A} \cap \tilde{B} = \emptyset$, and
- $\tilde{A} \cup \tilde{B} = \mathbb{R}^d \setminus \{z\}$,

if and only if

$$A \cap (\mathrm{conv}(B \cup \{z\})) = \emptyset \text{ and } B \cap (\mathrm{conv}(A \cup \{z\})) = \emptyset. \tag{1}$$

Proof of Claim 14. The direction $(\Rightarrow)$ is trivial, since if (w.l.o.g.) $B \cap (\mathrm{conv}(A \cup \{z\})) \neq \emptyset$ then there exist $a \in A$ and $b \in B$ such that b in contained in the open segment (a, z). But in this case, no point on the opposite ray $\{(1+\lambda)z - \lambda a : \lambda > 0\}$ can belong to $\tilde{A}$ or to $\tilde{B}$, a contradiction.

For the opposite direction, assume for the sake of convenience that $z = 0$ and that (1) is satisfied. An equivalent formulation of (1) is:

$$\text{Each ray from 0 meets at most one of the sets } A, B. \tag{2}$$

We shall now construct $\tilde{A}$ and $\tilde{B}$. First, let $A^+ = \bigcup_{\lambda>0}(\lambda A)$ (resp., $B^+ = \bigcup_{\lambda>0}(\lambda B)$) be the union of all open rays from 0 via A (resp., B). The sets A^+, B^+ are convex and $A \subset A^+, B \subset B^+$. Since each ray from the origin intersects A if and only if it intersects A^+, and similarly for B, the condition (2) is satisfied also for A^+, B^+. Therefore, A^+, B^+ are disjoint convex sets that are closed under addition and multiplication by a positive scalar. Let

$$\mathcal{D} = \{F \subset \mathbb{R}^d \setminus \{0\} : F \text{ is closed under addition and multiplication by a positive scalar}\},$$

and consider the family

$$\mathcal{F} = \{(A', B') : A', B' \in \mathcal{D}, A' \cap B' = \emptyset, A^+ \subset A', B^+ \subset B'\}.$$

Define a partial order on $\mathcal{F}$ by $(A', B') \leq (A'', B'')$ iff $A' \subseteq A''$ and $B' \subseteq B''$. Since each chain

$\{(A'_i, B'_i)\}$ is bounded from above by $(\cup_i A'_i, \cup_i B'_i)$, by Zorn's lemma, $\mathcal{F}$ contains a maximal element $(\tilde{A}, \tilde{B})$. Clearly $\tilde{A}, \tilde{B}$ are disjoint convex sets, $A \subset \tilde{A}, B \subset \tilde{B}$. It remains to prove that $\tilde{A} \cup \tilde{B} = \mathbb{R}^d \setminus \{0\}$. Assume on the contrary that there exists a point $w \in \mathbb{R}^d \setminus (\tilde{A} \cup \tilde{B} \cup \{0\})$. Then by the maximality of $(\tilde{A}, \tilde{B})$, the cone $\{\tilde{A} + \lambda w : \lambda > 0\}$ meets $\tilde{B} \cup \{0\}$. Therefore there exist $a \in \tilde{A}$ and $b' \in \tilde{B} \cup \{0\}$ such that $w + a = b'$. Similarly, there exist $b \in \tilde{B}$ and $a' \in \tilde{A} \cup \{0\}$ such that $w + b = a'$. Hence $b' - a = a' - b$ and it follows that $a + a' = b + b' \in \tilde{A} \cap \tilde{B}$, a contradiction. This completes the proof of Claim 14. ◁

Claim 14 implies the following consequence:

▶ **Corollary 15.** *Let $H \subset \mathbb{R}^d$ be a $(k+1)$-flat and let $J \subset H$ be a k-flat. Let $A, B \subset J$ be disjoint convex sets. Assume that J separates H into two open half-flats H^+, H^- and that $z \in H^+$. Then there exist two disjoint convex sets $\tilde{A}, \tilde{B}$ such that $A \subset \tilde{A}, B \subset \tilde{B}$ and $\tilde{A} \cup \tilde{B} = H \setminus \{z\}$ (see Figure 3). Moreover, the sets $\tilde{\tilde{A}} = (\tilde{A} \cap H^+) \cup A, \tilde{\tilde{B}} = (\tilde{B} \cap H^+) \cup B$ are disjoint convex sets such that $\tilde{\tilde{A}} \cap J = A, \tilde{\tilde{B}} \cap J = B$ and $(\tilde{\tilde{A}} \cup \tilde{\tilde{B}}) \cap H^+ = H^+ \setminus \{z\}$.*

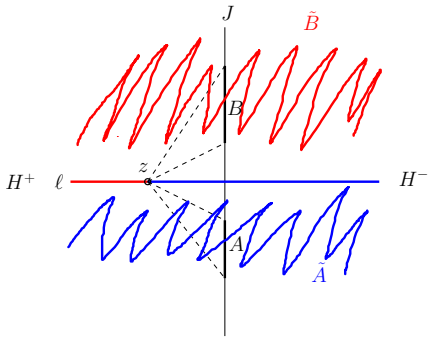


Figure 3 An illustration for Corollary 15 where $H = \mathbb{R}^2$ and J is a line. A and B are two segments, as illustrated in the figure. $\tilde{B}$ is the red half-open upper half-plane supported by ℓ, and $\tilde{A}$ is the blue half-open lower half-plane supported by ℓ.

Now we are ready to prove part (c) of Theorem 2. We prove the construction by induction on d. Given a construction in $\mathbb{R}^{d-2}$ of 3 convex sets that encapsulate $f(d-2)$ points, we extend each of the three convex sets to $\mathbb{R}^d$ in such a way that the previously encapsulated points are still encapsulated, and three more encapsulated points are formed. Since $f(d) = f(d-2) + 3$, such a construction completes the proof.

For the basic case, in $\mathbb{R}^0$ we can take $K_1 = K_2 = K_3 = \emptyset$, and in $\mathbb{R}^1$ the convex sets can be $K_1 = (-\infty, 0), K_2 = (0, 1)$ and $K_3 = (1, \infty)$. The construction for $d = 2$ is illustrated in Figure 4 (though it can be also obtained by the inductive argument).

In the induction step, let $J \subset \mathbb{R}^d$ be a $(d-2)$-flat. By the induction hypothesis there exist three pairwise disjoint convex sets $K'_1, K'_2, K'_3 \subset J$ such that $|J \setminus (K'_1 \cup K'_2 \cup K'_3)| = f(d-2)$. Let $\pi \subset \mathbb{R}^d$ be a plane orthogonal to J, hence $\pi \cap J$ is a point. W.l.o.g., assume that $\pi \cap J = \{0\}$. Consider 3 points $x_1, x_2, x_3 \in \pi$ equally distributed around 0 (see Figure 5). Consider the $(d-1)$-flat $H_1 = \mathrm{aff}(J, x_1)$, and let H_1^+ be the open half-flat of H_1 supported by J that contains x_1. (Formally, $H_1^+ = \{y + \lambda x_1 : y \in J, \lambda > 0\}$.) Define H_2^+, H_3^+ similarly.

We are supposed to produce three pairwise disjoint convex sets K_1, K_2, K_3, such that $K'_i \subset K_i$ for $i = 1, 2, 3$ and $\mathbb{R}^d \setminus (K_1 \cup K_2 \cup K_3) = J \setminus (K'_1 \cup K'_2 \cup K'_3) \cup \{x_1, x_2, x_3\}$. By Corollary 15, K'_1 and K'_2 can be extended to two disjoint convex sets K_1^3 and K_2^3 such that $K_1^3 \cap J = K'_1$, $K_2^3 \cap J = K'_2$ and $K_1^3 \cup K_2^3 = K'_1 \cup K'_2 \cup H_3^+ \setminus \{x_3\}$. Similarly, K'_1 and K'_3 can be extended to two disjoint convex sets K_1^2 and K_3^2 such that $K_1^2 \cap J = K'_1$, $K_3^2 \cap J = K'_3$ and

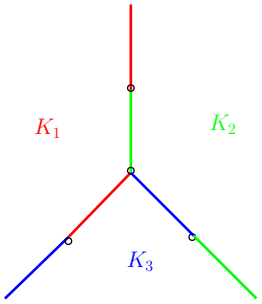


Figure 4 An illustration for the basic case $d = 2$ in the proof of Theorem 2(c).

$K_1^2 \cup K_3^2 = K_1' \cup K_3' \cup H_2^+ \setminus \{x_2\}$. K_2^1 and K_3^1 are obtained in the same way. Finally, we define $K_1 = K_1^2 \cup K_1^3 \cup \text{int}(\text{conv}(H_2^+ \cup H_3^+))$ and similarly, $K_2 = K_2^1 \cup K_2^3 \cup \text{int}(\text{conv}(H_1^+ \cup H_3^+))$ and $K_3 = K_3^1 \cup K_3^2 \cup \text{int}(\text{conv}(H_1^+ \cup H_2^+))$. The sets K_1, K_2 and K_3 encapsulate all the $f(d-2)$ points that were originally encapsulated by K_1', K_2', K_3' in J, and additionally encapsulate x_1, x_2, x_3. Therefore, we obtain $f(d-2)+3 = f(d)$ points that are encapsulated by 3 convex sets in $\mathbb{R}^d$. This completes the proof of part (c) of Theorem 2, and hence the proof of Theorem 2. ◀

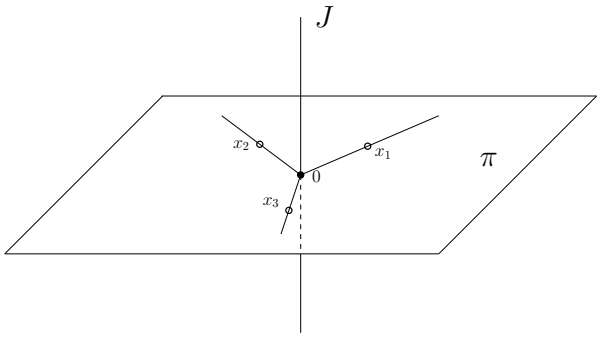


Figure 5 An illustration for the induction step in the proof of Theorem 2(c).

2.3 The number of points encapsulated by n convex sets in $\mathbb{R}^d$

Recall that $f(d,n)$ is the maximal number of points that can be encapsulated by n convex sets in $\mathbb{R}^d$. Clearly[3], for $d \geq 1$, $f(d,1) = 0, f(d,2) = 1$, and for all $n \geq 1$ we have $f(1,n) = n-1$. Define arbitrarily $f(0,n) = 1$. After obtaining the value $f(d,3) = \lfloor \frac{3d}{2} \rfloor + 1$, the following theorem yields a recursive upper bound on the function $f(d,n)$ in general.

▶ **Theorem 3** (Restatement). *For every $n > 3, d \geq 2$*

$$f(d,n) \leq \sum_{i=2}^{n-1} \left(\binom{n}{i} f(d,i) \right) + f(d-2,n). \tag{3}$$

Consequently,

$$f(d,n) \leq 2d^{n-2}n!. \tag{4}$$

[3] See also Theorem 9 below.

Proof. The proof of the recursive formula is by a double induction on n and d, where the basic cases $f(1,n)$, $f(d,2)$ and $f(d,3)$ have already been proved above. In the induction step, note that for each of the $\binom{n}{i}$ i-subsets of $K_1,\ldots,K_n$, the points of $\mathbb{R}^d \setminus \bigcup_{i=1}^n K_i$ that touch exactly this subset, are encapsulated by this subset, and by the induction hypothesis their number is bounded by $f(d,i)$. Therefore it remains to prove that the number of points that touch all n sets is at most $f(d-2,n)$.

Indeed, Let S' be the set of points that touch $K_1,\ldots,K_n$, and let $J = \mathrm{aff}(S')$. By Observation 11, $\dim(J) < d$. If $\dim(J) \le d-2$, we are done.

Otherwise, $\dim(J) = d-1$. In this case, as in the argument of Case 3 in the proof of Theorem 2, no point that is encapsulated by some subset of $K_1,\ldots,K_n$ lies outside J. Indeed, otherwise there exists $T \subset [n]$ and a point $p \in \mathbb{R}^d \setminus J$ that touches each K_i for $i \in T$ and no K_j for $j \notin T$. Then $\dim(\mathrm{conv}(S' \cup \{p\})) = d$, and since all the points in S' touch all the K_i's for $i \in T$, we have $\mathrm{conv}(S' \cup \{p\}) \subset \bigcap_{i\in T} \mathrm{cl}(K_i)$, and $\bigcap_{i\in T} K_i$ is d-dimensional. But then by Observation 11, p is not encapsulated by $\bigcup_{i\in T} K_i$.

Consequently, for every $T \subset [n]$, all the points that are encapsulated by $\bigcup_{i\in T} K_i$ are in J, and the bound $f(d-1,n)$, which is smaller than the right hand side of (3) by the induction hypothesis on d, follows.

The upper bound (4) follows from the recursive formula (3) by induction on n, for every fixed $d \ge 2$. The induction bases are the bounds $f(d,2) = 1$ and $f(d,3) \le \lfloor \frac{3d}{2} \rfloor + 1 \le 2d$, proved above. Assume that (4) holds for all $f(d,n')$, $n' < n$. Hence, by (3) we have

$$f(d,n) \le \sum_{i=2}^{n-1} \left(\binom{n}{i} f(d,i) \right) + f(d-2,n) \le \sum_{i=2}^{n-1} \left(\binom{n}{i} \cdot 2d^{i-2} i! \right) + 2(d-2)^{n-2} n!.$$

Define $h_{d,n}(i) = \binom{n}{i} \cdot 2d^{i-2} i!$. Note that for any $d \ge 2$, $n \ge 4$, $2 \le i \le n-2$,

$$\frac{h_{d,n}(i+1)}{h_{d,n}(i)} = \frac{\binom{n}{i+1} \cdot 2d^{i-1}(i+1)!}{\binom{n}{i} \cdot 2d^{i-2} i!} = \frac{n-i}{i+1} \cdot d(i+1) = (n-i)d \ge 2.$$

Hence, $\sum_{i=2}^{n-1} h_{d,n}(i) \le 2h_{d,n}(n-1)$. Thus, we have

$$\begin{aligned} f(d,n) &\le \sum_{i=2}^{n-1} \left(\binom{n}{i} \cdot 2d^{i-2} i! \right) + 2(d-2)^{n-2} n! \le 2nd^{n-3}(n-1)! + 2(d-2)^{n-2} n! \\ &= 2n! d^{n-2} \left(\tfrac{2}{d} + (\tfrac{d-2}{d})^{n-2} \right) \le 2n! d^{n-2}. \end{aligned}$$

This completes the proof of (4) by induction. ◀

We note that the upper bound (4) can be improved further with no much effort. However, it is not far from the best one can get from the recursion (3). Indeed, it is easy to see that for n constant, the dependence on d in (3) is $\Omega(d^{n-2})$, and that for d constant, the dependence on n in (3) is super-exponential.

3 Covering the complement of finite unions of convex sets by flats

In this section we study covering of $S = \mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$ by flats – i.e., affine subspaces of $\mathbb{R}^d$. The main result we prove is Theorem 6 which asserts that there exists a covering of S by $g(n,d)$ flats such that any $p \in S$ is covered by a flat whose dimension is $\mathrm{dm}(S,p)$.

3.1 Preliminaries

Recall that by Def. 5, the flat-dimension of S is $\mathrm{dm}(S) = \max\{k : S \text{ includes a } k\text{-simplex}\}$, and the local dimension of $p \in S$ is $\mathrm{dm}(S,p) = \min_{\epsilon>0} \mathrm{dm}(S \cap B(p,\epsilon))$.

Define $\mathrm{dm}(\emptyset) = -1$. Note that $\mathrm{dm}(S) = 0$ iff S is non-empty, but does not include a non-degenerate straight line segment, and $\mathrm{dm}(S) = d$ iff $\mathrm{int}(S) \neq \emptyset$. Note that for general sets $S \subset \mathbb{R}^d$, $\mathrm{dm}(S)$ does not necessarily coincides with the topological dimension of S, nor with the dimension of its affine hull, $\dim(\mathrm{aff}(S))$.

As for the local dimension, $\mathrm{dm}(S,p) = k$ if the intersection of S with every neighborhood of p includes a k-simplex, but the intersection of S with some neighborhood of p does not include a $(k+1)$-simplex. Refer to Figure 6 (and also to Figure 12 below) for an example of the local dimension. Note that if $\mathrm{dm}(S,p) = 0$ then there exists some neighborhood U of p such that $S \cap U$ contains no segment. In this case p is an isolated point of S, since by Theorem 6, S can be covered by finitely many points, hence $S \cap U$ is finite and therefore p is isolated.

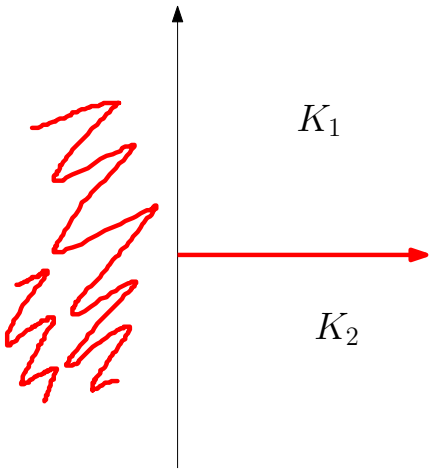


Figure 6 In the figure, the red-colored area is $S = \{(x,y) : x < 0\} \cup \{(x,0) : x \geq 0\} = \mathbb{R}^2 \setminus (K_1 \cup K_2)$, where $K_1 = \{(x,y) : x \geq 0, y > 0\}$ and $K_2 = \{(x,y) : x \geq 0, y < 0\}$. In this case, $\mathrm{dm}(S,(0,0)) = 2$ and $\mathrm{dm}(S,(1,0)) = 1$. The point $(1,0)$ is an ordinary point of S (with the x-axis as the corresponding 1-flat), while $(0,0)$ is not.

Ordinary points and ordinary flats.

▶ **Definition 16.** *The point p is a k-ordinary point of S ($p \in S \subset \mathbb{R}^d$) if*

1. $\mathrm{dm}(S,p) = k$.
2. *For some k-flat $H \subset \mathbb{R}^d$ and for some open neighborhood U of p, $S \cap U = H \cap U$.*

We call H a k-ordinary flat of S. Moreover, p is called an ordinary point of S if it is k-ordinary for some $0 \leq k \leq d$. Similarly for flats.

The set of k-ordinary points of S is a relatively open subset of S. Indeed, if p is a k-ordinary point of S and U, H are as in Definition 16, then every point $x \in S \cap U$ is k-ordinary as well with the same k-flat H. Note that if Definition 16 holds for U then it clearly holds for any smaller neighborhood $p \in U' \subset U$. Moreover, the flat H is determined by the point p, $H = \mathrm{aff}(U \cap S)$, for an appropriate neighborhood U as in Definition 16.

▶ Remark 17. If S' is relatively open subset of S, (i.e., $S' = S \cap U$ where U is an open subset of $\mathbb{R}^d$) then for every point $p \in S'$, $\mathrm{dm}(S,p) = \mathrm{dm}(S',p)$. Moreover, p is a k-ordinary point of S iff it is a k-ordinary point of S' (with the same k-ordinary flat.)

Strong covers.

▶ **Definition 18.** *Let $S \subset \mathbb{R}^d$. A collection $\mathcal{C}$ of flats in $\mathbb{R}^d$ is a strong cover of S, if:*

1. $|\mathcal{C}| < \infty$
2. *Each point $p \in S$ belongs to some k-flat $H \in \mathcal{C}$, where $k = \mathrm{dm}(S,p)$.*
3. *$\mathcal{C}$ is minimal, i.e., no proper subcollection of $\mathcal{C}$ satisfies (2).*

3.2 Proof of Theorem 6

Our main result in this section is the following restatement of Theorem 6 (where the value of the function g from the original statement follows from the proof).

▶ **Theorem 6** (Restatement). *Let $K_1, \ldots, K_n$ be convex subsets of $\mathbb{R}^d$ and let $S = \mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$. Then S admits a unique strong cover $\mathcal{C}$. Actually, $\mathcal{C}$ is the collection of all ordinary flats with respect to S.*

Note that p is encapsulated by $K_1, \ldots, K_n$ if and only if $\{p\}$ is an 0-ordinary flat w.r.t. $\mathbb{R}^d \setminus (K_1 \cup \ldots \cup K_n)$. Hence, both Theorems 2 and 3 above, supply quantitative bounds for special cases of Theorem 6.

It is easy to see that any strong cover $\mathcal{C}$ of S must contain all ordinary flats of S. Indeed, given a k-ordinary flat H, there exists some k-ordinary point $p \in S$ and some open neighborhood U of p, such that $S \cap U = H \cap U$. If $H \notin \mathcal{C}$ then consider all the k-flats $H' \in \mathcal{C}$. Since for any such H', $H \neq H'$, it follows that $\dim(H \cap H') < k$, therefore the neighborhood U is not covered by these (finitely many) k-flats H'. But as mentioned above, each point in $S \cap U$ is a k-ordinary point of S – a contradiction.

Therefore, in order to prove Theorem 6 we show that there are only finitely many ordinary flats with respect to S, and that condition (2) of Definition 16 is satisfied not only for the ordinary points of S (which is trivial), but also for the non-ordinary points of S.

Proof of Theorem 6. The proof is by a primary induction on d and a secondary induction on n. For the induction basis, let $p \in S \subset \mathbb{R}^d$ with $\mathrm{dm}(S,p) = k$. If $k = d$ then no inductive argument is needed since $p \in \mathrm{cl}(\mathrm{int}(S))$, and then the corresponding d-flat is $\mathbb{R}^d$. The case $d = 0$ is trivial. If $d = 1$ and $k = 0$ then p is an isolated point of S and the number of such points p is at most $n - 1$. The corresponding 0-flat in this case is clearly $\{p\}$. If $d = 1$ and $k = 1$ then $\mathbb{R}^1$ is the corresponding 1-flat. If $n = 1$ then for any $p \in S$, $\mathrm{dm}(S,p) = d$ and $\mathbb{R}^d$ is the d-ordinary flat.

For larger values of n and d we shall prove by induction the following:

1. For any $0 \leq k \leq d$, there are finitely many k-ordinary flats w.r.t. S.
2. For any $p \in S$ with $\mathrm{dm}(S,p) = k$ and for any $\epsilon > 0$, there is a k-ordinary point $q \in S$ such that $||p - q|| < \epsilon$.

▶ Remark 19. The second statement implies the existence of a k-ordinary flat J such that $p \in J$. Indeed, by the statement p is in the closure of the k-ordinary points of S, each of which is contained in a k-ordinary flat. But these flats form a closed set, hence p is contained in at least one of these k-ordinary flats.

The sets to which we apply the induction hypothesis are as follows:

- $S^i = \mathbb{R}^d \setminus \bigcup_{j \neq i} K_j$, $(i = 1, 2, \ldots, n)$, to which we apply the induction hypothesis with the same d and $n-1$ convex sets.
- For a flat H (to be specified later) with $\dim(H) < d$, consider $S^H = H \setminus (\bigcup_{j=1}^{n}(K_j \cap H))$, that is, the restriction[4] of the original system to H. Here we apply the induction hypothesis with the dimension $\dim(H)$.

We will prove that any k-ordinary flat w.r.t. S is either a k-ordinary flat w.r.t. some S^i or a k-ordinary flat for some S^H with $\dim(H) < d$. Hence we will be able to apply the induction hypothesis either to S^i or to S^H.

Let $p \in S$ with $\mathrm{dm}(S, p) = k$. There are two options:

- **Case 1:** For some $1 \leq i \leq n$, p does not touch K_i.
- **Case 2:** p touches all sets K_i, $1 \leq i \leq n$.

We handle each case separately.

Case 1: Since p does not touch K_i, it follows that $p \in \mathrm{int}(\mathbb{R}^d \setminus K_i)$, and thus, p has a positive distance from K_i. Let U be a neighborhood of p in $\mathbb{R}^d$ that satisfies $U \subset \mathrm{int}(\mathbb{R}^d \setminus K_i)$. Then $S \cap U = S^i \cap U$ and $\mathrm{dm}(S^i, p) = \mathrm{dm}(S, p) = k$. Moreover, for every corresponding k-flat J with $J \cap U = S \cap U$ we have $J \cap U = S^i \cap U$. Hence, the set of k-ordinary flats w.r.t. S that correspond to a point p as in case 1, is included in the union of the sets of k-ordinary flats w.r.t. S^i, $1 \leq i \leq n$, and we are done by the induction hypothesis on the S^i's.

Case 2: Let $G = \{p \in S : p \text{ touches all } K_i\text{'s}\}$ and let $H = \mathrm{aff}(G)$ be the flat that is spanned by G. If $H = \mathbb{R}^d$ then $\mathrm{conv}(G)$ is d-dimensional and $\mathrm{int}(\mathrm{conv}(G)) \subset \bigcap_{1 \leq i \leq n} K_i$. Then $\mathrm{dm}(p, S) = d$ holds for all $p \in G$, and the corresponding d-flat is $\mathbb{R}^d$. From now on we assume $\dim(H) < d$.

Clearly $\mathrm{dm}(S^H, p) \leq k$, and every neighborhood of p in S includes a k-simplex. We distinguish between two cases:

Case 2a: Any neighborhood U of p includes a k-simplex that is not included in H. In this case, in each sufficiently small neighborhood $U = B(p, \frac{1}{m})$ of p we can find a point q_m with $\mathrm{dm}(S, q_m) = k$ and $q_m \notin H$. By the definition of H, this means that for some $1 \leq i \leq n$, q_m does not touch K_i. By passing to a subsequence $\{q_{m_r}\}_{r=1}^{\infty}$ we can assume that for a fixed $1 \leq i \leq n$, each q_{m_r} does not touch K_i. By the induction hypothesis on S^i, each q_{m_r} is contained in a k-ordinary flat w.r.t. S^i, where the number of such flats is finite. By passing again to a subsequence, we can restrict ourselves to only one such k-ordinary flat, J. Therefore, we obtain in this flat a sequence of k-ordinary points that tend to p and by the closedness of J, $p \in J$.

Case 2b: There exists a neighborhood U of p such that every k-simplex in $S \cap U$ is contained in H. In this case, $\mathrm{dm}(S^H, p) = k$ and we can apply the induction hypothesis to S^H, since we have already assumed $\dim(H) < d$. By this induction hypothesis there exists a k-ordinary flat J' (w.r.t S^H) that contains p, and a sequence of k-ordinary points in $J' \cap S^H$ that tends to p.

The only thing that is left to prove is that J' is ordinary not only w.r.t. $S^H \subset H$ but also w.r.t. $S \subset \mathbb{R}^d$. Indeed, let $U' \subset U$ be a neighborhood of p such that $U' \cap J' = U' \cap S^H$. If some point $q \in U'$ is the limit of a sequence of points from S outside H, each point in the

[4] Note that along the proof, when considering S^H, we treat S^H as a subset of H (and not as a subset of $\mathbb{R}^d$).

sequence does not touch some K_i, and by passing to a subsequence, we can assume that all its elements do not touch K_i for a fixed i. Then, by the induction hypothesis on S^i, the whole sequence is contained in finitely many $(\leq k)$-flats, none of them is J'. The union of all these $(\leq k)$-flats, intersects J' in a $(<k)$-dimensional set, hence we can find $U'' \subset U'$ s.t. $U'' \cap J' = U'' \cap S$, namely, J' is k-ordinary also w.r.t. S. ◀

4 Bounding the number of ordinary hyperplanes

In this section we present the proof of Theorem 8 which essentially determines the maximal possible number of hyperplanes in the strong cover $\mathcal{C}$ of $S = \mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$, whose existence was proved in Theorem 6. In Theorem 20 we prove the assertion of the theorem for $d \geq 3$, and in Theorem 21 we prove it for $d = 2$. We formulate these theorems using the notion of *ordinary hyperplanes*, whose number is equal, by the proof of Theorem 6, to the notion ν_{d-1} used in the formulation of Theorem 8.

▶ **Theorem 20.** *Let $d \geq 3$ and let $S = \mathbb{R}^d \setminus (\cup_{i=1}^n K_i)$, where $\{K_i\}$ are convex sets. Then the number of ordinary hyperplanes (i.e., $(d-1)$-flats) w.r.t. S is at most $\binom{n}{2}$. On the other hand, the value $\binom{n}{2}$ is attained for some $K_1, \ldots, K_n$.*

Due to space limit, the proof is presented in [10, Theorem 4.1].

In the case $d = 2$, we prove an upper bound of $(\frac{3}{4} + o(1))\binom{n}{2}$, and provide a construction for which this value is obtained.

▶ **Theorem 21.** *Let $S = \mathbb{R}^2 \setminus (\cup_{i=1}^n K_i)$, where $\{K_i\}$ are convex sets. Then the maximum possible number of ordinary lines w.r.t. S is the Turán number $t(n,4) = (\frac{3}{4} + o(1))\binom{n}{2}$. On the other hand, this value is attained for some $K_1, \ldots, K_n$.*

Proof. For the upper bound we consider the geometric graph G, whose vertices are n arbitrary points $x_i \in \operatorname{int}(K_i)$ $(1 \leq i \leq n, \forall i \neq j : x_i \neq x_j)$, and whose edges are defined as follows: For each ordinary line ℓ that corresponds to a 1-ordinary point p_ℓ, there exist (as in the proof of Theorem 20) K_i, K_j with $\dim K_i = \dim K_j = 2$, such that ℓ is the only line that weakly separates K_i and K_j, $p \in \operatorname{cl}(K_i) \cap \operatorname{cl}(K_j)$. (Clearly, $\operatorname{int}(K_i) \cap \operatorname{int}(K_j) = \emptyset$.) We connect x_i to x_j in G by a two-edges polygonal path $[x_i, p_\ell, x_j]$.

By the disjointness of $\operatorname{int}(K_i)$ and $\operatorname{int}(K_j)$, the graph G (though not necessarily planar) does not include K_5, since otherwise there exist 5 pairwise disjoint open convex sets whose closures are pairwise intersecting, which is impossible since the restriction of G to the corresponding vertices is planar by the disjointness of the 5 $\operatorname{int}(K_i)$'s. Hence, by Turán's Theorem, $|E(G)| \leq t(n,4) = (\frac{3}{4} + o(1))\binom{n}{2}$, and the upper bound on the number of ordinary lines follows.

On the other hand, we present a construction that realizes every complete 4-partite graph K_{n_1,n_2,n_3,n_4} with $n_1 + n_2 + n_3 + n_4 = n$, in particular Turán's graph. The construction is a bit complicated, hence we present it in five steps accompanied by illustrations.

Step 1. First consider four closed convex sets K_1, K_2, K_3, K_4 as in Figure 7. Let $e_{ij} = K_i \cap K_j$ for $1 \leq i < j \leq 4$. Each segment e_{ij} on the line that separates K_i from K_j.

Our goal is to construct for each $1 \leq i \leq 4$, n_i open convex sets $K_i^1, \ldots, K_i^{n_i}$, each of which is a subset of a tiny perturbation of K_i. This construction has the property that for every $1 \leq i < j \leq 4$, and for every $1 \leq \ell_i \leq n_i, 1 \leq \ell_j \leq n_j$, there exists an ordinary line separating $K_i^{\ell_i}$ from $K_j^{\ell_j}$. By taking the n_i's $(1 \leq i \leq 4)$ as equal as possible and satisfying $n_1 + n_2 + n_3 + n_4 = n$, we obtain in this way n convex sets with $\Sigma_{i<j} n_i n_j = t(n,4)$ ordinary lines as asserted.

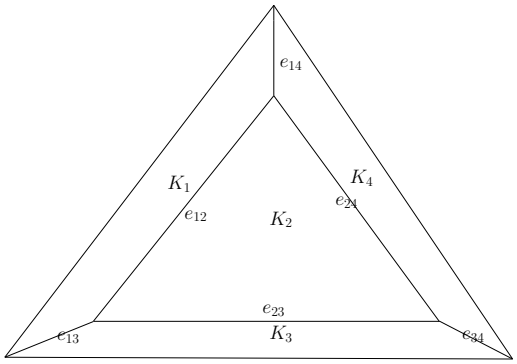


Figure 7 An illustration for step 1 in the proof of Theorem 21.

Step 2. For every $1 \leq i < j \leq 4$, draw a circular arc γ_{ij} through the endpoints of e_{ij} inside K_i (see Figure 8).

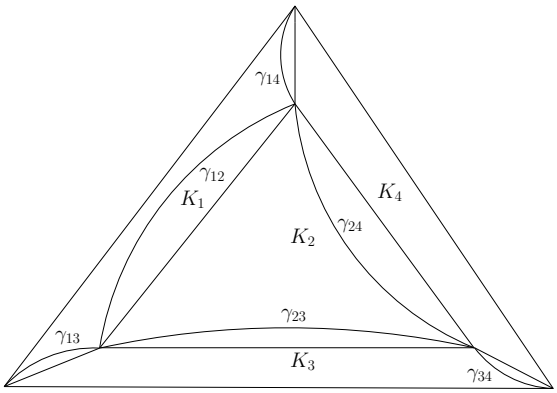


Figure 8 An illustration for step 2 in the proof of Theorem 21.

The radius of the circle associated with this arc, should be very large (in a sense to be clarified later). Note that in Figures 8-11 this radius is not large enough, just to make the illustration easier to follow.

Step 3. For every $1 \leq i < j \leq 4$, split γ_{ij} by $n_i + 1$ points into $n_i + 2$ parts. Let the edges of the polygonal path through these points (and the endpoints of e_{ij}) be $a_0^{ij}, a_1^{ij}, \ldots, a_{n_i+1}^{ij}$. See Figure 9, in which the polygonal paths corresponding to γ_{12}, γ_{23} and γ_{24} are illustrated. In this figure, $n_1 = 2$ and $n_2 = 3$.

Step 4. For every $1 \leq i < j \leq 4$, and $1 \leq t \leq n_i$, draw an arc δ_t^{ij} through the endpoints of a_t^{ij} to the side of K_j. Again, the radius of the circle associated with δ_t^{ij} should be very large (as will be determined later), much larger than in Figure 10 (in which still $n_1 = 2$ and $n_2 = 3$). Partition δ_t^{ij} into $n_j + 2$ sub-arcs by adding $n_j + 1$ points on it, and let the edges of the polygonal path through these points (and the endpoints of a_t^{ij}) be $b_{t,0}^{ij} \ldots, b_{t,n_j+1}^{ij}$. In Figure 11, these polygonal paths are illustrated, where $n_1 = n_3 = n_4 = 2$, and $n_2 = 3$.

Step 5. Let $1 \leq i \leq 4$. In this step we construct n_i open convex sets $K_i^1, \ldots, K_i^{n_i}$, each of which is a subset of a tiny perturbation of K_i. Each of these n_i convex sets will be the interior of the convex hull of several segments, and the radii of the circles mentioned in steps

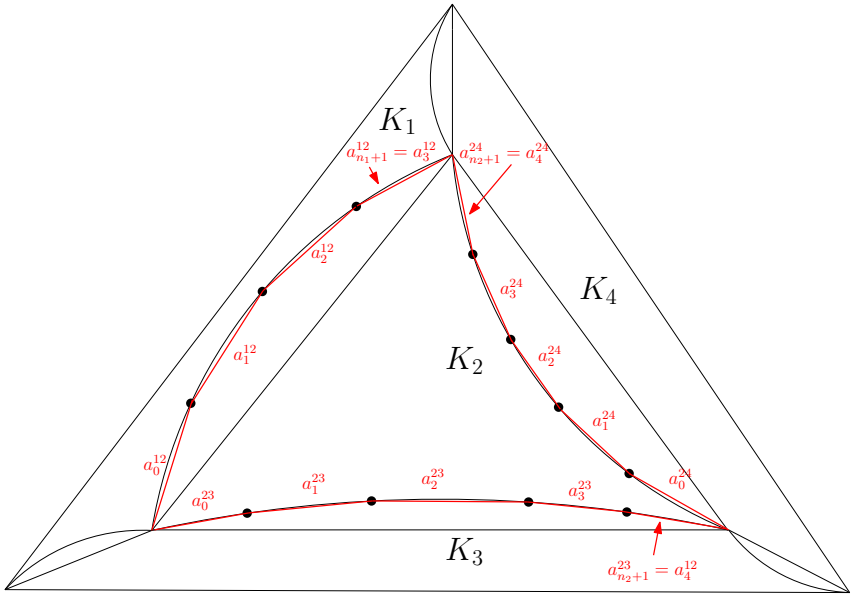


Figure 9 An illustration for step 3 in the proof of Theorem 21. Here $n_1 = 2, n_2 = 3$.

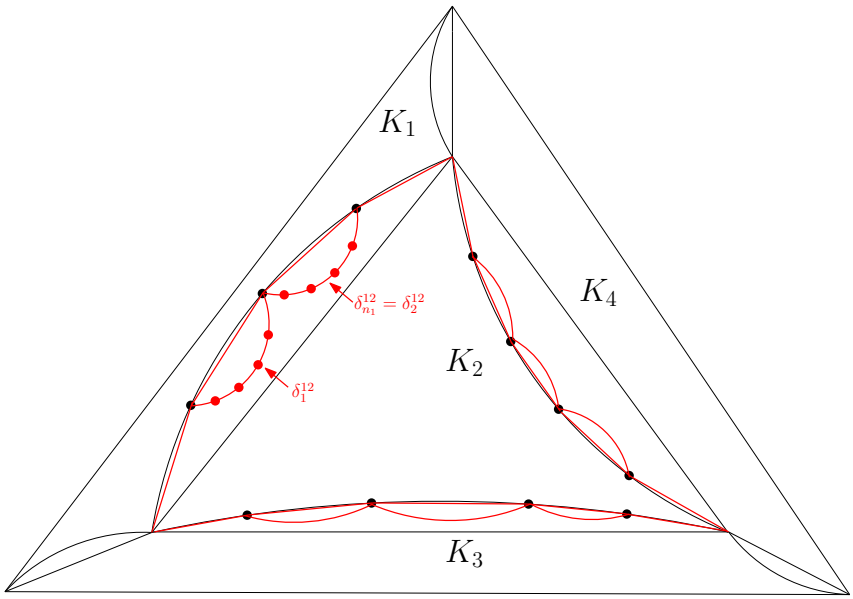


Figure 10 An illustration for step 4 in the proof of Theorem 21. Here $n_1 = n_3 = n_4 = 2, n_2 = 3$.

2 and 4 should be so large, such that all the segments that are involved in the construction of each K_i^ℓ $(1 \le \ell \le n_i)$ will be on the boundary of $\mathrm{cl}(K_i^\ell)$. Note that each K_i^ℓ is an open convex polygon[5].

[5] This is the reason why in the construction we omit the first and the last segments of each arc.

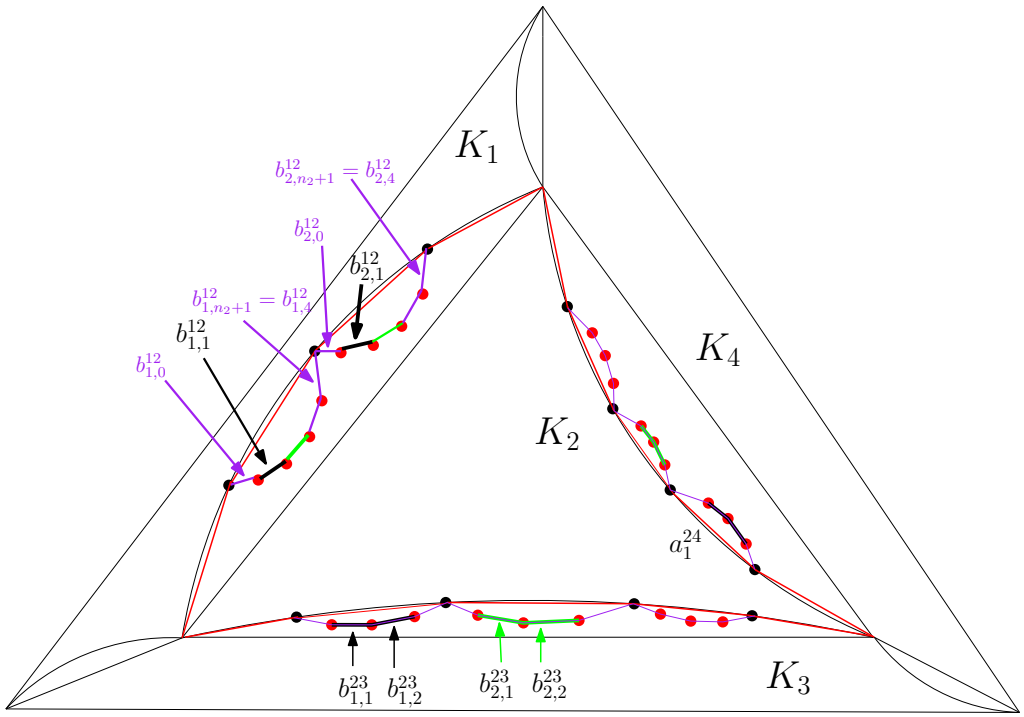


Figure 11 An illustration for steps 4-5 in the proof of Theorem 21. Here $n_1 = n_3 = n_4 = 2, n_2 = 3$.

The first such set, K^1_i, is the interior of the convex hull of the following segments:

$$\begin{cases} b^{ij}_{1,1}, b^{ij}_{2,1}, \ldots b^{ij}_{n_j,1} & : 1 \le j < i \\ b^{ij}_{1,1}, b^{ij}_{1,2}, \ldots b^{ij}_{1,n_j} & : i < j \le 4 \end{cases}$$

In Figure 11 (that still demonstrates the case where $n_1 = 2, n_2 = 3$), the segments that form K^1_2 are colored black. Roughly speaking, from the arcs inside K_i we take consecutive "b" edges whose endpoints lie on the first "δ"-arc corresponding to each edge of K_i, and from the arcs out of K_i that correspond to an edge of K_i, we take the first "b"-edge from each "δ"-arc. Recall that the radii of all arcs involved are so large, that (unlike the drawing in Figures 8 -11) all these "b"-edges that form each K^ℓ_i are in convex position.

The second such set, K^2_i, is the interior of the convex hull of the following segments, colored by green in Figure 11:

$$\begin{cases} b^{ij}_{1,2}, b^{ij}_{2,2}, \ldots b^{ij}_{n_j,2} & : 1 \le j < i \\ b^{ij}_{2,1}, b^{ij}_{2,2}, \ldots b^{ij}_{2,n_j} & : i < j \le 4 \end{cases}$$

In general, for each $1 \le \ell \le n_i$, the set K^ℓ_i is the interior of the convex hull of the following segments:

$$\begin{cases} b^{ij}_{1,\ell}, b^{ij}_{2,\ell}, \ldots b^{ij}_{n_j,\ell} & : 1 \le j < i \\ b^{ij}_{\ell,1}, b^{ij}_{\ell,2}, \ldots b^{ij}_{\ell,n_j} & : i < j \le 4 \end{cases}$$

In this way, as was mentioned just after Step 1, we construct n open convex sets with $\Sigma_{i<j} n_i n_j$ distinct ordinary lines. This completes the proof of Theorem 21. ◀

5 The complement of the union of two convex sets

In this section we present the proof of Theorem 9, which fully characterizes the set of possible vectors $(v_0, \ldots, v_d)$ attainable as the numbers of ordinary k-flats of $S = \mathbb{R}^d \setminus (K_1 \cup K_2)$, where $K_1, K_2 \subset \mathbb{R}^d$ are convex sets. The theorem follows immediately from Claims 22, 23 below.

A construction for $d = 2$, in which a strong cover of S consists of exactly one k-ordinary flat for every $0 \leq k \leq d$ (or in other words, a construction which corresponds to the vector $(v_0, v_1, v_2) = (1, 1, 1)$) is presented in Figure 12.

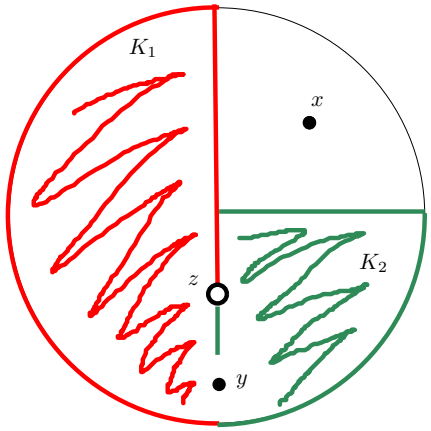


Figure 12 The circle in this figure is centered at the origin. The sets K_1 and K_2 are colored with red and green, respectively. The set K_1 contains the left half circle (including its arc) and the segment $[(0, 1), (0, -\frac{1}{3}))$, and K_2 contains the quarter right-bottom circle and the segments $[(0, 0), (1, 0)]$ and $((0, -\frac{1}{3}), (0, -\frac{2}{3})]$. The point $x \in S = \mathbb{R}^2 \setminus (K_1 \cup K_2)$ is of local dimension $d = 2$ and the corresponding 2-ordinary flat is $\mathbb{R}^2$. The point y is of local dimension 1 and the corresponding 1-ordinary flat is the y-axis. The point z is of local dimension 0 and the corresponding 0-ordinary flat is $\{z\}$ itself.

The following claim implies that for every $0 \leq k \leq d$, a strong cover of S contains at most one k-ordinary flat.

▷ Claim 22. Let $K_1, K_2 \subset \mathbb{R}^d$ be convex sets, and let $S = \mathbb{R}^d \setminus (K_1 \cup K_2)$. If J_1, J_2 are k-ordinary flats w.r.t. S then $J_1 \subset J_2$ or $J_2 \subset J_1$.

Due to space limit, the proof of Claim 22 is presented in [10, Claim 5.1].

On the other hand, for any subset T of $\{0, 1, \ldots, d\}$ we can find $K_1, K_2 \subset \mathbb{R}^d$ such that $\mathcal{C}$ contains a k-flat if and only if $k \in T$.

▷ Claim 23. Let $f : \{0, \ldots, d\} \to \{0, 1\}$. Then there exist convex sets $K_1, K_2 \subset \mathbb{R}^d$ such that the number of k-ordinary flats in a strong cover of $S = \mathbb{R}^d \setminus (K_1 \cup K_2)$ is $f(k)$.

The proof of Claim 23 is presented in [10, Claim 5.2].

References

1 P. K. Agarwal, J. Pach, and M. Sharir. State of the union, of geometric objects: A review. In *Proc. Joint Summer Research Conf. on Discrete and Computational Geometry: 20 Years Later, Contemp. Math. 452*, pages 9–48. AMS, 2008.

2 B. Aronov, O. Cheong, M. G. Dobbins, and X. Goaoc. The number of holes in the union of translates of a convex set in three dimensions. *Discret. Comput. Geom.*, 57(1):104–124, 2017. doi:10.1007/S00454-016-9820-4.

3 B. Aronov and M. Sharir. The common exterior of convex polygons in the plane. *Comput. Geom.*, 8:139–149, 1997. doi:10.1016/S0925-7721(96)00004-1.

4 B. Aronov and M. Sharir. On translational motion planning of a convex polyhedron in 3-space. *SIAM J. Comput.*, 26(6):1785–1803, 1997. doi:10.1137/S0097539794266602.

5 X. Bei, N. Chen, and S. Zhang. Solving linear programming with constraints unknown. In *ICALP 2015, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2015. doi:10.1007/978-3-662-47672-7_11.

6 A. Björner and G. Kalai. An extended Euler–Poincaré theorem. *Acta Math.*, 161:279–303, 1988.

7 J. Cibulka, M. Korbelář, J. Kynčl, V. Mészáros, R. Stolař, and P. Valtr. On three measures of non-convexity. *Israel J. Math.*, 218:331–369, 2017.

8 H. Edelsbrunner and J. Pach. Maximum Betti numbers of Čech complexes. In *SoCG 2024*, volume 293 of *LIPIcs*, pages 53:1–53:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SOCG.2024.53.

9 G. O. H. Katona. On a problem of L. Fejes Tóth. *Stud. Sci. Math. Hung.*, 12(1–2):77–80, 1977.

10 Chaya Keller and Micha A. Perles. Complements of finite unions of convex sets, arxiv:2508.19413, 2025. arXiv:2508.19413.

11 M. D. Kovalev. A property of convex sets and its application. *Mat. Zametki (in Russian)*, 44:89–99, 1988.

12 J. Lawrence and W. D. Morris Jr. Finite sets as complements of finite unions of convex sets. *Discret. Comput. Geom.*, 42(2):206–218, 2009. doi:10.1007/S00454-009-9184-0.

13 J. Matoušek and P. Valtr. On visibility and covering by convex sets. *Israel J. Math.*, 113(3):341–379, 1999.

Computing the Bottleneck Distance Between Persistent Homology Transforms

Michael Kerber
Institute of Geometry, Graz University of Technology, Austria

Elena Xinyi Wang
Institute of Geometry, Graz University of Technology, Austria
University of Fribourg, Switzerland

Abstract

The Persistent Homology Transform (PHT) summarizes a shape in $\mathbb{R}^m$ by collecting persistence diagrams obtained from linear height filtrations in all directions on $\mathbb{S}^{m-1}$. It enjoys strong theoretical guarantees, including continuity, stability, and injectivity. A natural way to compare two PHTs is to use the bottleneck distance between their diagrams as the direction varies. Prior work has either compared PHTs by sampling directions or, in 2D, computed the exact *integral* of bottleneck distance over all angles via a kinetic data structure. We improve the integral objective to $\tilde{O}(n^5)$ in place of the earlier $\tilde{O}(n^6)$ bound, where n denotes the number of simplices. For the *max* objective, we give an $\tilde{O}(n^3)$ expected-time algorithm in $\mathbb{R}^2$ and an $\tilde{O}(n^5)$ expected-time algorithm in $\mathbb{R}^3$.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Theory of computation → Computational geometry

Keywords and phrases Kinetic data structure, bottleneck distance, persistent homology transform, vineyards

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.62

Related Version *Full Version*: https://arxiv.org/pdf/2512.00821

Funding *Michael Kerber*: Supported by the Austrian Science Fund (FWF) 10.55776/P33765.
Elena Xinyi Wang: Supported by the Austrian Science Fund (FWF) 10.55776/P33765.

1 Introduction

Persistent homology captures the multiscale evolution of homology classes along a filtration induced by a real-valued function on a space. It has proved useful across fields such as materials science [9, 14], computational biology [12, 19], as well as machine learning [3, 13], providing robust descriptors for high-dimensional data. A persistence diagram is a concise multiset summary of persistent homology, recording the birth–death times of features. Similarity between objects is then assessed by comparing their diagrams, most commonly via the bottleneck distance. Computationally, this reduces to a minimum-bottleneck perfect matching on a bipartite graph with diagonal copies, and algorithms for this problem have been studied extensively in both theory and practice [2, 6, 10, 15, 16].

Since a single diagram depends on the chosen filter and may miss geometry, the Persistent Homology Transform (PHT) collects diagrams from all directions to retain fuller information [20]. It assigns to a shape in $\mathbb{R}^m$ the collection of persistence diagrams obtained from linear height filtrations over all directions on $\mathbb{S}^{m-1}$. It possesses many desirable theoretical properties such as continuity, stability, and injectivity [5, 11, 20]. These properties make the PHT an effective topological descriptor for applications. However, unlike other directional-transform–based signatures such as the Euler Characteristic Transform, the PHT lacks efficient comparison algorithms, limiting its practical use. A natural way to compare PHTs is to sample directions and compute the bottleneck distance only at those angles. This

© Michael Kerber and Elena Xinyi Wang;
licensed under Creative Commons License CC-BY 4.0

inevitably leads to information loss. To integrate the bottleneck distance over the parameter domain (e.g., $\mathbb{S}^{m-1}$), prior work by Munch, Wang, and Wenk introduced a kinetic "hourglass" approach that maintains matchings as the parameter moves. It computes this integral with an $\tilde{O}(n^6)$ bound for data in $\mathbb{R}^2$ [18]. In this work, we focus on the max variant, which we define as the supremum of the bottleneck distance over all directions.

Our Contributions. We provide an $\tilde{O}(n^3)$ expected-time algorithm to compute the max bottleneck distance between PHTs of shapes in $\mathbb{R}^2$ and an $\tilde{O}(n^5)$ expected-time algorithm in $\mathbb{R}^3$, where $\tilde{O}$ means the logarithmic factor is ingored. Additionally, we improve the computation of the integral distance from the previous $\tilde{O}(n^6)$ bound to $\tilde{O}(n^5)$ in $\mathbb{R}^2$.

For the max bottleneck distance, we use techniques from computational geometry and topology and inspired by the strategy of Bjerkevik and Kerber for computing the matching distance for two-parameter modules [1]. At a high level, the solution for 2D and 3D are similar. We first reformulate the problem as a decision problem: given a threshold λ, we determine whether the max bottleneck distance is at most λ. To solve this, we sweep the direction parameter around the sphere while maintaining the bottleneck matching between the two persistence diagrams. The key insight is that this matching changes only at discrete "events": points where persistence diagrams change combinatorially or where edge weights cross the threshold λ. At each event, we update the matching using a single augmenting path search in the geometric bipartite graph, exploiting the fact that only local changes occur.

We then identify that the maximum bottleneck distance must occur at one of a finite set of candidate values, corresponding to critical points and intersections of the difference functions between simplices. While there are $O(n^4)$ candidates in 2D and $O(n^6)$ in 3D, we avoid exhaustive enumeration through a randomized incremental construction. We maintain a band containing the true maximum and iteratively refine it by processing simplices in random order, testing whether their associated candidates lie within the current band. The randomization ensures that we expect only $O(\log n)$ refinements, reducing the expected complexity to $\tilde{O}(n^3)$ in 2D and $\tilde{O}(n^5)$ in 3D.

Outline. We begin in Section 2 with a brief overview of persistent homology, persistence diagrams, and the PHT, and we formalize the problem. Section 3 presents our $\mathbb{R}^2$ algorithm: first at a high level, then with full technical details. The improvement for the integral distance is shown in Section 4. Section 5 extends the approach to $\mathbb{R}^3$ before concluding in Section 6.

2 Preliminary

2.1 Persistent homology

Persistent homology is a multi-scale summary of the connectivity of objects in a nested sequence of subspaces; see [7, 8] for an introduction. For the purposes of this section, we define a *persistence diagram* to be a finite collection of points $\{(b_i, d_i)\}_i$ with $d_i \geq b_i$ for all i.

Given two persistence diagrams X and Y, a partial matching is a bijection $\eta : X' \to Y'$ on a subset of the points $X' \subseteq X$ and $Y' \subseteq Y$; we denote this by $\eta : X \rightleftharpoons Y$. The cost of a partial matching is the maximum over the L_∞-norms of all pairs of matched points and the distance between the unmatched points to the diagonal:

$$c(\eta) = \max\left(\{\|x - \eta(x)\|_\infty \mid x \in X'\} \cup \{\tfrac{1}{2}|z_2 - z_1| \mid (z_1, z_2) \in (X \setminus X') \cup (Y \setminus Y')\}\right) \quad (1)$$

and the bottleneck distance is defined as $d_B(X, Y) = \inf_{\eta : X \rightleftharpoons Y} c(\eta)$.

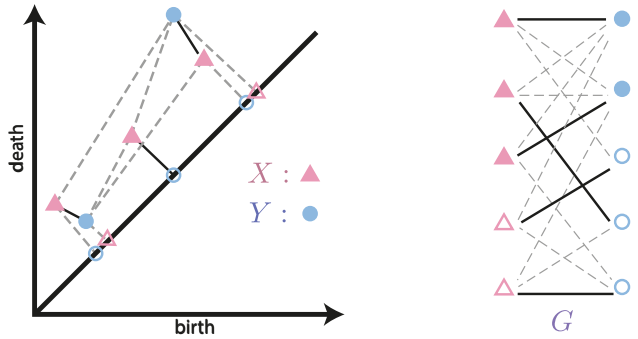


Figure 1 Construction of the bipartite graph G based on the persistence diagrams X and Y.

We can reduce finding the bottleneck distance between persistence diagrams to the problem of finding the bottleneck cost of a bipartite graph. Let X and Y be two persistence diagrams given as finite lists of off-diagonal points. For any off-diagonal point $z = (z_1, z_2)$, the orthogonal projection to the diagonal is $z' = ((z_1 + z_2)/2, (z_1 + z_2)/2)$. Let $\overline{X}$ (resp. $\overline{Y}$) be the set of orthogonal projections of the points in X (resp. Y). Set $U = X \sqcup \overline{Y}$ and $V = Y \sqcup \overline{X}$. We define the complete bipartite graph $G = (U \sqcup V, U \times V, \mathrm{c})$, where for $u \in U$ and $v \in V$, the weight function c is given by

$$\mathrm{c}(uv) = \begin{cases} \|u - v\|_\infty & \text{if } u \in X \text{ or } v \in Y \\ 0 & \text{if } u \in \overline{X} \text{ and } v \in \overline{Y}. \end{cases}$$

An example of the bipartite graph construction is shown in Figure 1. This graph can be used to compute the bottleneck distance of the input diagrams because of the following lemma.

▶ **Lemma 1** (Reduction Lemma [8]). *For the above construction of G, $d_B(G) = d_B(X, Y)$.*

2.2 Persistent homology transform

Let K be a finite simplicial complex embedded in $\mathbb{R}^m$. For any direction $\omega \in \mathbb{S}^{m-1}$, the height function $h_\omega : |K| \to \mathbb{R}$ is defined by $h_\omega(x) = \langle x, \omega \rangle$. The lower-star filtration induced on the abstract complex K is given by:

$$h_\omega(\sigma) = \max\{h_\omega(v) \mid v \in \sigma\},$$

and the sublevelset at height a is the full subcomplex induced by vertices $\{v \in K \mid h_\omega(v) \leq a\}$. Filtering K by h_ω produces a persistence diagram $\mathrm{Dgm}_k(h_\omega^K)$. We focus on $k = 0, 1$ for $m = 2$, and $k = 0, 1, 2$ for $m = 3$.

The *persistent homology transform (PHT)* is defined as:

$$\mathrm{PHT}(K) : \mathbb{S}^{m-1} \to \mathcal{D}, \quad \omega \mapsto \mathrm{Dgm}(h_\omega^K),$$

where $\mathcal{D}$ is the space of persistence diagrams. The PHT is injective [20, 5], making it a faithful shape representation.

2.3 Maximum bottleneck distance

In this work, we are interested in comparing two finite embedded simplicial complexes, $K, K' \in \mathbb{R}^m$, $m = 2$ or 3, by comparing their PHTs. We define the *maximum bottleneck distance* between K and K' to be:

$$d_\infty(\mathrm{PHT}(K), \mathrm{PHT}(K')) = \max_{\omega \in \mathbb{S}^{m-1}} d_B\left(\mathrm{Dgm}(h_\omega^K), \mathrm{Dgm}(h_\omega^{K'})\right).$$

This distance is a variant of a similar integral distance that has been studied previously by Munch, Wang, and Wenk [18]:

$$d_1(\mathrm{PHT}(K), \mathrm{PHT}(K')) = \int_{\mathbb{S}^{m-1}} d_B\left(\mathrm{Dgm}(h_\omega^K), \mathrm{Dgm}(h_\omega^{K'})\right) d\omega.$$

Both d_∞ and d_1 inherit stability from the bottleneck distance. However, they differ from a computational perspective. Computing d_1 requires full knowledge of the whole domain, which is inevitably expensive. Our d_∞ metric seeks the maximum value, which allows us to reformulate the problem as finding the supremum over a finite set of candidate values rather than tracking the bottleneck distance continuously.

A key property that makes our algorithmic approach feasible is that the bottleneck distance function is Lipschitz continuous, which is a direct consequence of the stability of the bottleneck distance (see the full version for a complete proof):

▶ **Lemma 2.** *Let $H(\omega) := d_B\big(\mathrm{Dgm}(h_\omega^K), \mathrm{Dgm}(h_\omega^{K'})\big)$ denote the bottleneck distance between persistence diagrams. Then the function $H : \mathbb{S}^{m-1} \to \mathbb{R}$ is Lipschitz continuous with constant $R_K + R_{K'}$, where $R_K = \max_{v \in K^{(0)}} \|v\|$ and $R_{K'} = \max_{v \in (K')^{(0)}} \|v\|$.*

3 Computing d_∞ in $\mathbb{R}^2$

Our setup in this section is as follows: we are given two geometric simplicial complexes K, K' in $\mathbb{R}^2$ with at most n simplices each. The goal is to compute d_∞: we first introduce the necessary notation, then give a high-level overview, and finally present the technical details.

3.1 Events

For each simplex $\alpha \in K \cup K'$, we define an insertion curve $I_\alpha(\omega)$. These curves record the height at which each simplex enters the sublevel-set filtration induced by direction $\omega \in S^1$. Explicitly, if the k-simplex α has vertices $\{v_0, \ldots, v_k\} \subseteq V(K)$, its insertion curve is

$$I_\alpha(\omega) := \max_{0 \le i \le k} h_\omega(v_i), \quad h_\omega(x) = \langle x, \omega \rangle.$$

Each I_α is a piecewise-trigonometric function on S^1.

There are at most $2n$ such curves. We now define *difference curves* that capture both matched and unmatched contributions to the bottleneck distance. For distinct simplices α, σ:

$$\Delta_{\alpha,\sigma}(\omega) := \begin{cases} |I_\alpha(\omega) - I_\sigma(\omega)| & \text{if } \alpha, \sigma \text{ are from different complexes} \\ \frac{1}{2}|I_\alpha(\omega) - I_\sigma(\omega)| & \text{if } \alpha, \sigma \text{ are from the same complex} \end{cases}$$

These are real-valued functions on S^1; geometrically, one can picture their graphs as curves in $S^1 \times \mathbb{R}_{\ge 0}$. The factor of $\frac{1}{2}$ corresponds to the contribution of unmatched points to the bottleneck distance, see Equation 1. There are $O(n^2)$ such curves for $|K|, |K'| = O(n)$. We call the vertex v_i achieving $\max_{0 \le j \le k} \langle v_j, \omega \rangle$ the *active vertex* of α at ω. As ω varies, the active vertex changes at discrete directions where two vertices achieve the same height. Within any interval where the active vertices remain fixed, the difference curve simplifies to either $|\langle u - w, \omega \rangle|$ (for different complexes) or $\frac{1}{2}|\langle u - w, \omega \rangle|$ (for same complex), which is a simple trigonometric function. See Figure 2 for an example. The insertion and difference curves are both well-behaved. See Figure 3 for an example.

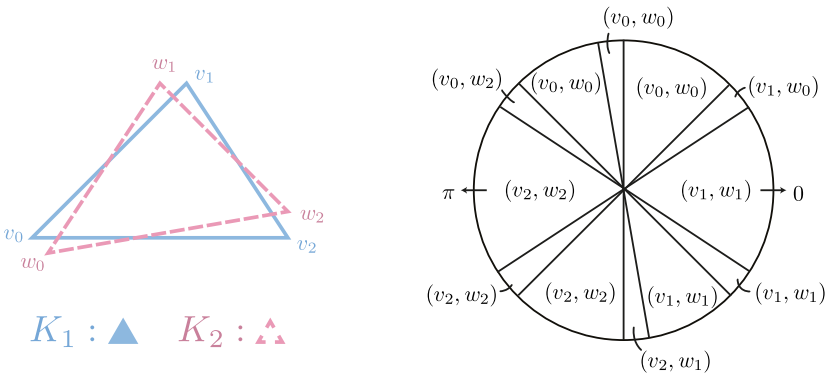


Figure 2 On the left are two simplices with vertices $v_{0,1,2}$ and $w_{0,1,2}$ respectively. On the right is $\mathbb{S}^1$ partitioned into regions with the same active vertex pairs (v_i, w_j).

▶ **Lemma 3.** *Each difference curve $\Delta_{\alpha,\sigma}$ crosses any horizontal line $y = \lambda$ at most $O(1)$ times as ω varies over $[0, 2\pi]$. Since there are $O(n^2)$ difference curves in total, this yields $O(n^2)$ crossings with any horizontal line. In particular, there are $O(n^2)$ insertion-curve intersections, as these correspond to difference curves crossing the line $y = 0$.*

The bound on insertion-curve intersections follows immediately since we have $O(n)$ curves total, giving at most $\binom{O(n)}{2} = O(n^2)$ pairwise intersections. For threshold crossings, each difference curve $\Delta_{\alpha,\sigma}$ is piecewise trigonometric with $O(1)$ pieces (as simplices in $\mathbb{R}^2$ have at most 3 vertices), and crosses any threshold λ at most $O(1)$ times. With $O(n^2)$ pairs total, this yields $O(n^2)$ threshold crossings. The detailed geometric analysis is in the full version.

In each direction ω, the bottleneck distance is realized by the value of a difference curve $\Delta_{\alpha,\sigma}(\omega)$. When this maximum is unique in a neighborhood of ω, the continuity of the bottleneck function $H(\omega) := d_B\left(\mathrm{Dgm}(h^K_\omega), \mathrm{Dgm}(h^{K'}_\omega)\right)$ ensures that the same curve or persistence value realizes the bottleneck distance in an ε-neighborhood. We call the points in S^1 where the realizing curve or value changes locally *events*, and the values H attains at events are the *candidate values*.

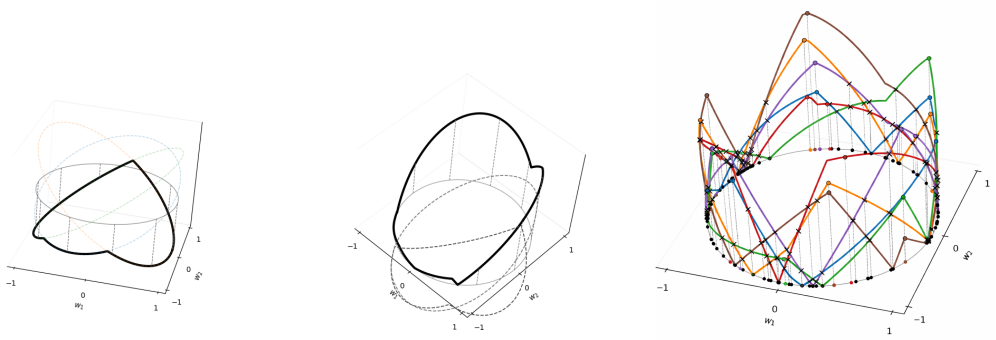
▶ **Theorem 4** (Events for d_∞ in $\mathbb{R}^2$)**.** *The direction ω^* attaining d_∞ occurs at either*

1. *a local maximum of a difference curve $\Delta_{\alpha,\sigma}$, or*
2. *an intersection of two difference curves. See Figure 3 for an illustration.*

Proof. Let ω^* be the direction that attains d_∞. By Lemma 2, if the bottleneck distance at ω^* is realized by a unique difference curve in a neighborhood, then ω^* must be a local maximum of that difference curve, giving case (1). This includes both matched pairs (when $\alpha \in K, \sigma \in K'$) and unmatched point persistence (when α, σ are from the same complex). Otherwise, at least two curves achieve the same maximum value at ω^*, giving case (2). Note that intersections of insertion curves (where the persistence diagram changes combinatorially) form a special case of (2), as when two insertion curves I_α, I_β from the same complex coincide, all difference curves of the form $\Delta_{\alpha,\sigma}$ and $\Delta_{\beta,\sigma}$ are equal. ◀

▶ Remark 5. When we refer to "intersections" of difference curves, we mean isolated transverse crossings, not intervals of coincidence. Endpoints of overlaps also count as intersections, since the realizing curve may change at those boundaries.

▶ **Corollary 6.** *There are $O(n^4)$ candidate values where d_∞ may be attained.*



(a) (b) (c)

Figure 3 (a) An insertion curve $I_\alpha(\omega)$. (b) A difference curve $\Delta_{\alpha,\sigma}(\omega)$. (c) Arrangement of four difference curves; local maxima and pairwise intersections mark the candidate values of Theorem 4.

Proof. There are $O(n^2)$ difference curves: $O(n^2)$ from pairs $(\alpha, \sigma) \in K \times K'$, plus $O(n^2)$ from pairs within the same complex. Each has $O(1)$ local maxima (see the full version for the explicit bound), contributing $O(n^2)$ type-(1) candidates. For type-(2) candidates, each pair of difference curves intersects at $O(1)$ heights, giving $\binom{O(n^2)}{2} = O(n^4)$ intersections total. Thus the total number of candidates is $O(n^4)$. ◀

▶ Remark 7. When simplices share vertices, their difference curves may partially coincide. For instance, if $\alpha, \beta \in K$ share vertex v and $\sigma, \tau \in K'$ share vertex w, then $\Delta_{\alpha,\sigma}$ and $\Delta_{\beta,\tau}$ coincide in regions where v and w are simultaneously active. Despite these overlaps, each difference curve retains a unique identity at the combinatorial level through its defining simplex pair. In particular, our candidate enumeration remains valid: if the maximum bottleneck distance is achieved at direction ω^*, then at least one combinatorially distinct difference curve must realize this maximum, and this curve's critical points and intersections are in our candidate set. The full technical detail is in the full version.

3.2 Overview of the algorithm

We describe the high-level structure of our algorithm next. On this level, our approach is similar to the algorithm of Bjerkevik and Kerber to compute the matching distance [1]. However, the approaches differ in the technical realization of the predicates which are described in the subsequent subsections.

Decision procedure. Given a threshold $\lambda \geq 0$, we decide whether $d_\infty \leq \lambda$ by sweeping ω around $\mathbb{S}^1$ and maintaining a bottleneck matching. As ω rotates, we update the persistence diagrams at $O(n^2)$ insertion-curve intersections and perform single augmenting-path updates whenever an edge weight crosses the threshold. This decision procedure runs in $\tilde{O}(n^3)$ time.

Basic algorithm. By Theorem 4, d_∞ is attained at one of $O(n^4)$ candidate values corresponding to intersections of difference curves and their local maxima. These candidates can be enumerated and sorted in $O(n^4 \log n)$ time. Combined with binary search using our decision procedure, this yields an $O(n^4 \log n + n^3 \operatorname{polylog}(n)) = \tilde{O}(n^4)$ algorithm.

Improved algorithm via band refinement. By Theorem 4, there are $O(n^4)$ total candidates where d_∞ could be attained. To achieve $\tilde{O}(n^3)$ time, we avoid enumerating all candidates.

We call a candidate an α-candidate if it arises from a difference curve $\Delta_{\alpha,\sigma}$ for some σ (when $\alpha \in K$), from $\Delta_{\tau,\alpha}$ for some τ (when $\alpha \in K'$), or from $|I_\alpha(\omega) - I_\beta(\omega)|/2$ for some β in the same complex as α.

We maintain a height band $(\lambda_1, \lambda_2]$ satisfying $d_\infty > \lambda_1$ and $d_\infty \leq \lambda_2$. We process simplices from both complexes in random order. For each simplex α, we apply:

1. Existence predicate: Tests if any α-candidate lies in $(\lambda_1, \lambda_2]$ in $O(n^2 \operatorname{polylog}(n))$ time. If no candidate exists in the band, we skip to the next simplex.
2. Band-refinement predicate: If the existence test is positive, we enumerate all $O(n^3)$ α-candidates in the band and refine via binary search in $O(n^3 \operatorname{polylog}(n))$ time.

The resulting total running time is $O(n \cdot n^2 \operatorname{polylog}(n) + C \cdot n^3 \operatorname{polylog}(n))$, where C is the number of band-refinement calls. As we will show, processing the simplices in a random order ensures $\mathbb{E}[C] = O(\log n)$, yielding expected time $\tilde{O}(n^3)$.

3.3 Decision problem

We now describe how to decide whether $d_\infty \leq \lambda$ for a given threshold $\lambda \geq 0$. The key idea is to sweep the direction ω around S^1 and check if the bottleneck distance ever exceeds λ.

By Lemma 1, at each direction ω, the bottleneck distance d_B equals the cost of a minimum-bottleneck matching in a bipartite graph $G(\omega)$ constructed from the two persistence diagrams $X = \operatorname{Dgm}(h^K_\omega)$ and $Y = \operatorname{Dgm}(h^{K'}_\omega)$ with diagonal copies. Define the subgraph $G_\lambda(\omega)$ to contain only those edges of $G(\omega)$ whose weights are $\leq \lambda$. Then $d_B(X, Y) \leq \lambda$ if and only if $G_\lambda(\omega)$ admits a perfect matching.

Our strategy is to maintain a perfect matching in $G_\lambda(\omega)$ as ω sweeps from 0 to 2π. If at any point no perfect matching exists, we conclude $d_\infty > \lambda$; otherwise, if the sweep completes with a valid matching at every direction, then $d_\infty \leq \lambda$.

Sweep algorithm. We initialize at $\omega = 0$ by computing a perfect matching in $G_\lambda(0)$ using Hopcroft-Karp in $O(n^{2.5})$ [15]; if none exists, $d_\infty > \lambda$.

As ω increases from 0 to 2π, the bipartite graph $G_\lambda(\omega)$ changes only at discrete events:

- Diagram changes: At intersections of insertion curves, points move to or from the diagonal in the persistence diagrams, updating the bipartite graph structure.
- Threshold crossings: When a difference curve $\Delta_{\alpha,\sigma}(\omega)$ crosses the threshold λ, the corresponding edge enters or leaves $G_\lambda(\omega)$.

Maintaining the matching. A matching M is maintained across the sweep. At each event:

- If a new edge enters $G_\lambda(\omega)$ (weight decreases below λ), M remains valid.
- If an edge leaves $G_\lambda(\omega)$ (weight exceeds λ) and is not in M, then M remains valid.
- If an edge $(u, v) \in M$ becomes invalid, Hall's theorem guarantees that a new perfect matching exists if and only if there is an augmenting path from the newly exposed vertices. We search for such a path in $O(n \log n)$ time using the geometric structure [10]. If found, we augment along this path to restore a perfect matching. If no augmenting path exists, then $G_\lambda(\omega)$ has no perfect matching, so $d_B > \lambda$ at this ω, implying $d_\infty > \lambda$.

At diagram change events (insertion curve intersections), we update the persistence diagrams using the vineyard algorithm [4]. When two insertion curves I_α and I_β from the same complex intersect, the relative order of exactly two simplices changes in the filtration. Following Bjerkevik and Kerber [1], this corresponds to a single transposition in the barcode pairing, which can be updated in $O(n)$ time.

If the sweep completes with a valid perfect matching at every ω, then $d_\infty \leq \lambda$.

▶ **Lemma 8.** *The decision procedure runs in $\tilde{O}(n^3)$ time.*

Proof. At each event, we either update the diagram structure in $O(n)$ time (vineyard update for insertion curve intersections) or search for a single augmenting path in $O(n \log n)$ time (for threshold crossings). With $O(n^2)$ total events from Lemma 3, the total time is $\tilde{O}(n^3)$. ◀

3.4 Candidate pruning via band refinement

Recall that we have $O(n^4)$ total candidates where d_∞ may be attained. To find d_∞ efficiently, we maintain a height band $(\lambda_1, \lambda_2]$ containing d_∞ and iteratively shrink it by processing simplices from both complexes in random order.

We distinguish between two types of intersections between difference curves. *Intra-intersections* occur when two difference curves share a common simplex: either Δ_{α,σ_1} and Δ_{α,σ_2} sharing simplex α, or $\Delta_{\beta_1,\tau}$ and $\Delta_{\beta_2,\tau}$ sharing simplex τ. *Inter-intersections* occur when $\Delta_{\alpha,\sigma}$ and $\Delta_{\beta,\tau}$ meet with all four simplices distinct: $\alpha \neq \beta$ and $\sigma \neq \tau$.

Pre-refinement. Before processing individual simplices, we refine the band to exclude certain events from its interior: let S be the set containing all local maxima of difference curves $\Delta_{\alpha,\sigma}$ for all pairs (including both different-complex and same-complex pairs), all intra-α intersections where $\Delta_{\alpha,\sigma_1}(\omega) = \Delta_{\alpha,\sigma_2}(\omega)$ for some $\omega \in \mathbb{S}^1$ and distinct σ_1, σ_2, and all intra-τ intersections where $\Delta_{\beta_1,\tau}(\omega) = \Delta_{\beta_2,\tau}(\omega)$ for distinct β_1, β_2.

Since there are $O(n^2)$ difference curves, each with $O(1)$ local maxima, we have $O(n^2)$ maxima total. For intra-intersections, each of the $O(n)$ simplices in K contributes $\binom{O(n)}{2} = O(n^2)$ pairs of curves that could intersect, with $O(1)$ intersection points each, giving $O(n^3)$ intra-α intersections. Similarly for intra-τ intersections. Thus $|S| = O(n^3)$.

We binary search with the decision procedure to find the largest value in S below d_∞ and the smallest at or above it, setting these as λ_1 and λ_2. The resulting band $(\lambda_1, \lambda_2]$ contains no intra-events in its interior. This takes $O(n^3 \operatorname{polylog} n)$ time.

3.4.1 Predicate 1: Existence test

After pre-refinement, we process simplices to detect inter-intersections.

▶ **Lemma 9.** *For fixed $\alpha \in K$, determining whether any inter-intersection involving α lies in $(\lambda_1, \lambda_2]$ takes $O(n^2 \log n)$ time.*

Proof. We seek intersections of the form $\Delta_{\alpha,\sigma} \cap \Delta_{\beta,\tau}$ where $\beta \neq \alpha$. For each $\tau \in K'$, we define the *red curves* $R_\alpha = \{\Delta_{\alpha,\sigma} : \sigma \neq \alpha\}$ as all curves involving simplex α, and the *blue curves* $B_\tau = \{\Delta_{\beta,\tau} : \beta \neq \tau\}$ as all curves involving simplex τ. (See Figure 2 for the geometric setting; the red and blue curves are real-valued functions on S^1, and we detect their intersection as real-valued crossing events while sweeping around S^1.)

The key insight from pre-refinement is that within the band $(\lambda_1, \lambda_2]$, no two red curves can intersect (these would be intra-α intersections), and no two blue curves can intersect (these would be intra-τ intersections).

Each difference curve $\Delta_{\alpha,\sigma}(\omega)$ consists of $O(1)$ trigonometric pieces, switching between different expressions at directions where the active vertex changes. Since simplices in $\mathbb{R}^2$ have at most 3 vertices, there are $O(1)$ such transition points. When we clip to the band $[\lambda_1, \lambda_2]$, each curve crosses the boundary at $O(1)$ points, subdividing into $O(1)$ monotone arcs within the band.

We apply the standard Bentley-Ottmann sweep-line algorithm to all $O(n)$ arcs. Since red arcs do not intersect each other and blue arcs do not intersect each other (by pre-refinement), any intersection must be between a red and blue arc. We terminate immediately upon detecting the first such intersection. This takes $O(n \log n)$ time per choice of τ. Testing all $O(n)$ choices of $\tau \in K'$ takes $O(n^2 \log n)$ total time for fixed α. ◀

3.4.2 Predicate 2: Band refinement

When the existence test detects an inter-intersection, we must enumerate all candidates involving α to refine the band.

▶ **Lemma 10.** *When an inter-intersection involving α is detected, we can enumerate all $O(n^3)$ α-candidates in $(\lambda_1, \lambda_2]$ and compute a refined band with no α-candidates in its interior in $O(n^3 \operatorname{polylog} n)$ time.*

Proof. After pre-refinement, only inter-intersections remain in the band's interior. For simplex α, there are $O(n^3)$ such intersections (choosing σ, β, τ with $O(n)$ choices each). We enumerate these candidates, sort them, and refine the band using binary search with the decision procedure, taking $O(n^3 \operatorname{polylog} n)$ time. ◀

▶ **Theorem 11.** *After global pre-refinement, processing each simplex α takes $O(n^2 \log n + \mathbb{1}_\alpha n^3 \operatorname{polylog} n)$ total time, where $\mathbb{1}_\alpha$ is the indicator function of the existence test.*

Proof. This follows directly from Lemma 9 and 10. ◀

3.4.3 Randomized analysis

▶ **Theorem 12.** *Processing simplices from both K and K' in uniformly random order yields $\mathbb{E}[C] = O(\log n)$ refinements, where C counts positive existence tests.*

Proof. For each simplex α, define x_α as the smallest α-candidate that is $\geq d_\infty$ and y_α as the largest α-candidate that is $< d_\infty$. Since d_∞ is realized by at least one candidate, for some simplex α^* we have $x_{\alpha^*} = d_\infty$. Every interval $(y_\alpha, x_\alpha]$ contains d_∞, so our maintained band always has the form $(y_{\alpha_i}, x_{\alpha_j}]$ for some processed simplices α_i, α_j.

When processing simplices in random order, we obtain random permutations of the sequences $\{x_\alpha\}$ and $\{y_\alpha\}$. Simplex α passes the existence test and the refinement is triggered if and only if there exists an α-candidate in the current band. This occurs when x_α is smaller than all previously seen $x_{\alpha'}$ values (making it a left-to-right minimum), or y_α is larger than all previously seen $y_{\alpha'}$ values (making it a left-to-right maximum).

Indeed, if x_α is not a minimum, then some previously processed α' satisfies $x_{\alpha'} \leq x_\alpha$, so the band's upper bound is already at most x_α, excluding all α-candidates with value $\geq x_\alpha$. Likewise, if y_α is not a maximum, the band's lower bound already exceeds y_α, excluding all α-candidates with value $\leq y_\alpha$. Thus, when neither x_α nor y_α improves the band boundaries, every α-candidate lies outside (y_α, x_α), and none can fall in the current band.

By a classical result on random permutations [17]: in a uniformly random ordering of n distinct values, the i-th element is a left-to-right minimum with probability $1/i$ (being the smallest of the first i elements). By linearity of expectation, the expected number of left-to-right minima is $H_n = \sum_{i=1}^{n} \frac{1}{i} = O(\log n)$. The same bound holds for maxima. Therefore, the expected number of refinements is at most the expected number of minima in $\{x_\alpha\}$ plus the expected number of maxima in $\{y_\alpha\}$, giving $\mathbb{E}[C] \leq O(\log n) + O(\log n) = O(\log n)$. ◀

▶ **Corollary 13.** *The candidate pruning algorithm runs in expected time*

$$O(n \cdot n^2 \log n + \log n \cdot n^3 \operatorname{polylog}(n)) = \tilde{O}(n^3).$$

Proof. We process $O(n)$ simplices from both complexes. Each requires an $O(n^2 \log n)$ existence test. In expectation, only $O(\log n)$ trigger the $O(n^3 \operatorname{polylog}(n))$ band refinement. The refinement cost dominates, giving $\tilde{O}(n^3)$ expected time. ◀

4 Computing d_1 in $\mathbb{R}^2$

Our framework also improves the computation of the integral bottleneck distance d_1 from the previous $\tilde{O}(n^6)$ bound [18] to $\tilde{O}(n^5)$.

The integral requires maintaining the bottleneck matching as ω sweeps around $\mathbb{S}^1$. As ω varies, the bottleneck distance $H(\omega)$ changes only at the $O(n^4)$ difference curve intersection events identified in Section 3. At each event, we update the matching in $O(n \log n)$ time via an augmenting path search to identify which difference curve realizes the new bottleneck. Between consecutive events at angles ω_i and ω_{i+1}, the bottleneck distance is given by a single difference curve that can be integrated analytically.

Thus the integral distance can be computed in $\tilde{O}(n^5)$ total time: $O(n^4)$ events, each requiring $O(n \log n)$ processing. This improves upon the hourglass data structure approach of [18], which maintained two kinetic heaps and required $\tilde{O}(n^6)$ time.

5 Computing d_∞ in $\mathbb{R}^3$

We now consider embedded complexes in $\mathbb{R}^3$. Directions range over the sphere $\mathbb{S}^2$, difference curves become surfaces, and intersections occur along curves rather than at points. The algorithm follows the same framework as 2D, but requires traversing an arrangement on $\mathbb{S}^2$ via an Euler tour rather than sweeping a circle. This yields expected complexity $\tilde{O}(n^5)$.

5.1 Events

Let $K, K' \subset \mathbb{R}^3$ be finite simplicial complexes with at most n simplices each. For each simplex $\alpha \in K \cup K'$, $\omega \in S^2$, the *insertion surface* is:

$$I_\alpha : S^2 \to \mathbb{R}, \quad I_\alpha(\omega) := \max_{v \in \alpha} \langle v, \omega \rangle$$

Each I_α is the upper envelope of at most 4 trigonometric forms on S^2. The loci where the maximizing vertex changes are great circles $\{\omega \in S^2 : \langle v_i - v_j, \omega \rangle = 0\}$.

For distinct simplices α, σ, the *difference surface* is a real-valued function on $\mathbb{S}^2$:

$$\Delta_{\alpha,\sigma}(\omega) := \begin{cases} |I_\alpha(\omega) - I_\sigma(\omega)| & \text{if } \alpha, \sigma \text{ are from different complexes} \\ \frac{1}{2}|I_\alpha(\omega) - I_\sigma(\omega)| & \text{if } \alpha, \sigma \text{ are from the same complex} \end{cases}$$

Within each region where the active vertices are fixed, $\Delta_{\alpha,\sigma}$ simplifies to $|\langle u - w, \omega \rangle|$ for active vertices $u \in \alpha$ and $w \in \sigma$. Two difference surfaces $\Delta_{\alpha,\sigma}$ and $\Delta_{\beta,\tau}$ intersect along curves on $\mathbb{S}^2$ where $\Delta_{\alpha,\sigma}(\omega) = \Delta_{\beta,\tau}(\omega)$.

▶ **Lemma 14.** *For simplicial complexes in $\mathbb{R}^3$:*

1. *Each pair of insertion surfaces intersects along $O(1)$ great circle arcs on $\mathbb{S}^2$.*
2. *Each pair of difference surfaces intersects along $O(1)$ great circle arcs on $\mathbb{S}^2$.*
3. *Each intersection curve between difference surfaces has $O(1)$ local maxima.*
4. *Each difference surface crosses any threshold λ along $O(1)$ curves on $\mathbb{S}^2$.*

The proof follows from the piecewise structure: since simplices in $\mathbb{R}^3$ have at most 4 vertices, each insertion surface has $O(1)$ regions of fixed active vertex. Within each region, the surfaces are trigonometric forms and their intersections are great circles. The detailed geometric analysis is in the full version.

▶ **Theorem 15** (Events for d_∞ in $\mathbb{R}^3$). *The direction ω^* attaining d_∞ occurs at one of:*

1. *A local maximum of a difference surface $\Delta_{\alpha,\sigma}$*
2. *A local maximum along an intersection curve on $\mathbb{S}^2$ where two difference surfaces meet*
3. *A triple intersection point where three difference surfaces meet*

Proof. The bottleneck distance function $H(\omega)$ is continuous on S^2. At any $\omega^* \in S^2$ where d_∞ is attained, consider the difference surfaces achieving the maximum value $H(\omega^*)$.

If a unique surface $\Delta_{\alpha,\sigma}$ achieves this maximum in a neighborhood of ω^*, then by continuity, ω^* must be a local maximum of that surface (case 1). This covers both matched pairs and unmatched persistence. If exactly two surfaces achieve the maximum at ω^* but no other surface does so nearby, then ω^* lies on their intersection curve on $\mathbb{S}^2$. Since H is locally determined by these two surfaces along the curve, ω^* must be a local maximum along this curve (case 2). If three or more surfaces achieve the maximum at ω^*, then generically three surfaces meet at isolated points, giving case 3. ◀

▶ **Corollary 16.** *There are $O(n^6)$ candidate values where d_∞ may be attained.*

Proof. We count candidates by type. Type 1 has $O(n^2)$ difference surfaces (including both between-complex and within-complex pairs), each having $O(1)$ maxima, contributing $O(n^2)$ candidates. Type 2 consists of $\binom{O(n^2)}{2} = O(n^4)$ pairs with $O(1)$ maxima per intersection curve, giving $O(n^4)$ candidates. Type 3 is $\binom{O(n^2)}{3} = O(n^6)$ triple intersection points. This gives us the total of $O(n^6)$ candidates. ◀

5.2 Overview of algorithm

Decision procedure. Given threshold λ, we decide whether $d_\infty \leq \lambda$ by traversing an arrangement on $\mathbb{S}^2$. Similar to the 2D case, we track two types of events:

- Diagram changes: $O(n^2)$ intersection curves between insertion surfaces where the persistence diagrams change combinatorially;
- Threshold crossings: $O(n^2)$ curves on $\mathbb{S}^2$ where difference surfaces cross the threshold λ.

Overlaying these curves yields an arrangement of $O(n^4)$ complexity on $\mathbb{S}^2$. We traverse the dual of this arrangement via an Euler tour by traversing each edge twice, maintaining a bottleneck matching throughout. Each diagram change requires $O(n)$ time to update the barcode pairing using the vineyard algorithm, while each threshold crossing may require $O(n \log n)$ time for augmenting path search. The total complexity is $\tilde{O}(n^5)$.

Improved algorithm via band refinement. We maintain a band $(\lambda_1, \lambda_2]$ containing d_∞ and process simplices in random order. For each simplex α:

1. Existence predicate: Test if any α-candidate lies in the band. Following the approach from 2D adapted to surfaces: for fixed α and $\sigma \in K'$, consider the surface $\Delta_{\alpha,\sigma}$. For each $\beta \in K \setminus \{\alpha\}$, compute intersection curves with all $\Delta_{\beta,\tau}$ surfaces. Within the band, these $O(n)$ curves form disjoint arcs on $\mathbb{S}^2$ (no intra-β intersections after pre-refinement). Similarly get $O(n)$ disjoint arcs from surfaces involving $\gamma \in K \setminus \{\alpha, \beta\}$. Check for red-blue intersections in $O(n \log n)$ time. Testing all combinations takes $O(n^4 \log n)$.

2. Band refinement: If positive, enumerate all α-candidates in the band: $O(n^5)$ triple intersections, $O(n^3)$ curve maxima, $O(n)$ surface maxima. Then refine via binary search in $\tilde{O}(n^5)$ time.

Using the same randomization analysis as 2D, we expect $O(\log n)$ refinements. This results in total expected complexity of $O(n \cdot n^4 \log n + \log n \cdot n^5 \operatorname{polylog} n) = \tilde{O}(n^5)$.

5.3 Decision problem

Given threshold $\lambda \geq 0$, we decide whether $d_\infty \leq \lambda$ by constructing and traversing an arrangement on $\mathbb{S}^2$. Unlike the 2D case where we sweep around a circle, here we must systematically explore the entire sphere.

We first construct an arrangement $\mathcal{A}$ on $\mathbb{S}^2$ consisting of:

- $O(n^2)$ curves from insertion surface intersections (where persistence diagrams change)
- $O(n^2)$ curves where difference surfaces equal λ (threshold crossings)

Each insertion surface is piecewise trigonometric with $O(1)$ pieces, so pairs intersect along $O(1)$ curves each. Similarly, each difference surface crosses the threshold λ along $O(1)$ curves. The overlay of all these curves forms an arrangement with $O(n^4)$ vertices, edges, and faces.

▶ **Theorem 17.** *Whether $d_\infty \leq \lambda$ can be decided in $\tilde{O}(n^5)$ time.*

Proof. We construct the arrangement $\mathcal{A}$ and compute an Euler tour of its dual graph, treated as a planar graph embedded on $\mathbb{S}^2$. The tour visits each edge twice, once in each direction.

Starting from an arbitrary face with a perfect matching in $G_\lambda(\omega)$, we traverse the tour and maintain the matching throughout. At insertion curve crossings where the persistence diagram changes, we update the barcode pairing via a single transposition in $O(n)$ time using the vineyard algorithm. At threshold crossings where an edge weight exceeds λ, we search for an augmenting path in $O(n \log n)$ time. If no augmenting path exists, we conclude $d_\infty > \lambda$ and terminate. If the tour completes with valid matchings maintained throughout, then $d_B(\omega) \leq \lambda$ for all $\omega \in \mathbb{S}^2$, hence $d_\infty \leq \lambda$.

The arrangement has $O(n^4)$ edges in total. The $O(n^4)$ diagram changes contribute $O(n^5)$ total time at $O(n)$ each, while the $O(n^2)$ threshold crossings contribute $\tilde{O}(n^3)$ at $O(n \log n)$ each. However, the bottleneck arises from traversing all $O(n^4)$ edges of the arrangement, where at each edge crossing we may need to update the matching. Since each edge crossing potentially requires $O(n \log n)$ time for augmenting path search, the total complexity is $O(n^4 \cdot n \log n) = \tilde{O}(n^5)$. ◀

5.4 Candidate pruning via band refinement and complexity analysis

As in the 2D case, we maintain a band $(\lambda_1, \lambda_2]$ containing d_∞ and process simplices in random order. We first apply a pre-refinement to exclude all intra-events from the band.

Pre-refinement. Before processing individual simplices, we refine the band to exclude all intra-events from its interior. We call a triple intersection point $\Delta_{\alpha,\sigma} \cap \Delta_{\beta,\tau} \cap \Delta_{\gamma,\rho}$ an *intra-event* if any two of the six involved simplices are equal. Let S be the set containing all $O(n^2)$ local maxima of difference surfaces, all $O(n^4)$ local maxima along surface intersection curves, and all intra-event triple intersections. Since there are $O(n^5)$ ways to choose 5 distinct simplices and $O(1)$ ways to make two of them equal, we have $|S| = O(n^5)$. We use binary search with the decision procedure to find the largest value in S that is $< d_\infty$ (setting λ_1) and the smallest value in S that is $\geq d_\infty$ (setting λ_2), yielding the band $(\lambda_1, \lambda_2]$ that contains d_∞ but excludes all values of S from its interior. This takes $O(n^5 \operatorname{polylog} n)$ time.

▶ **Lemma 18.** *For fixed $\alpha \in K$, we can test whether any inter-intersection involving α lies in the band $(\lambda_1, \lambda_2]$ in $O(n^4 \log n)$ time.*

Proof. Fix $\alpha \in K$ and $\sigma \in K'$, giving the difference surface $\Delta_{\alpha,\sigma}$. We seek triple intersection points of the form $\Delta_{\alpha,\sigma} \cap \Delta_{\beta,\tau} \cap \Delta_{\gamma,\rho}$ where $\beta, \gamma \in K \setminus \{\alpha\}$ and $\tau, \rho \in K'$, with the intersection value lying in $(\lambda_1, \lambda_2]$.

For fixed $\beta \in K \setminus \{\alpha\}$, compute the intersection curves between $\Delta_{\alpha,\sigma}$ and all surfaces $\Delta_{\beta,\tau}$ for $\tau \in K'$. Each pair of surfaces intersects along great circle arcs on $\mathbb{S}^2$ where $\Delta_{\alpha,\sigma}(\omega) = \Delta_{\beta,\tau}(\omega)$. We obtain $O(n)$ intersection curves on $\mathbb{S}^2$.

The crucial observation is that when we restrict to the band $\{\omega : \lambda_1 < \Delta_{\alpha,\sigma}(\omega) \leq \lambda_2\}$, these $O(n)$ curves become disjoint arcs. Any intersection between curves $\Delta_{\alpha,\sigma} \cap \Delta_{\beta,\tau_1}$ and $\Delta_{\alpha,\sigma} \cap \Delta_{\beta,\tau_2}$ would yield a triple point where $\Delta_{\alpha,\sigma} = \Delta_{\beta,\tau_1} = \Delta_{\beta,\tau_2}$. This is an intra-event (two surfaces share β), excluded from the band's interior via pre-refinement.

Similarly, for fixed $\gamma \in K \setminus \{\alpha, \beta\}$, the intersection curves between $\Delta_{\alpha,\sigma}$ and all $\Delta_{\gamma,\rho}$ surfaces yield another set of $O(n)$ disjoint arcs within the band. We now have two collections of arcs on the restricted domain of $\Delta_{\alpha,\sigma}$: the "red" arcs from β-surfaces and the "blue" arcs from γ-surfaces. A red-blue intersection corresponds to a triple point $\Delta_{\alpha,\sigma} \cap \Delta_{\beta,\tau} \cap \Delta_{\gamma,\rho}$, which is an inter-intersection involving α.

To detect red-blue intersections, we apply stereographic projection from the north pole of $\mathbb{S}^2$ to map the band region to a planar domain and use a sweep-line algorithm. As shown in the full version, the projection preserves the disjointness of red and blue arc families while mapping great circle arcs to well-behaved curves (circles and lines) that intersect $O(1)$ times. The sweep-line algorithm thus detects all red-blue intersections in $O(n \log n)$ time.

Iterating over all choices of $\sigma \in K'$, $\beta \in K \setminus \{\alpha\}$, and $\gamma \in K \setminus \{\alpha, \beta\}$ gives total complexity $O(n \cdot n \cdot n \cdot n \log n) = O(n^4 \log n)$. ◀

▶ **Lemma 19.** *When the existence test is positive for simplex α, we can enumerate all $O(n^5)$ candidates involving α that lie in $(\lambda_1, \lambda_2]$.*

Proof. For each α, the pre-refinement step excludes the $O(n)$ local maxima of difference surfaces and $O(n^2)$ local maxima of intersection curves. We only need to consider triple intersection points $\Delta_{\alpha,\sigma} \cap \Delta_{\beta,\tau} \cap \Delta_{\gamma,\rho}$: with $O(n)$ choices each for $\sigma, \beta, \tau, \gamma, \rho$, and $O(1)$ intersection points per triple, we get $O(n^5)$ candidates. Total enumeration takes $O(n^5)$ time. Binary search with the decision procedure then refines the band in $\tilde{O}(n^5)$ time. ◀

▶ **Theorem 20** (Total complexity in 3D)**.** *The algorithm runs in expected time $\tilde{O}(n^5)$.*

Proof. We process $O(n)$ simplices from both complexes. For each simplex α, we perform the existence test in $O(n^4 \log n)$ time. If positive, we enumerate and refine in $\tilde{O}(n^5)$ time. We expect $O(\log n)$ refinements using the same analysis as in 2D. The total expected complexity is therefore $O(n \cdot n^4 \log n + \log n \cdot n^5 \operatorname{polylog} n) = \tilde{O}(n^5)$. ◀

6 Conclusion

We have presented the first algorithms for computing the maximum bottleneck distance between Persistent Homology Transforms, achieving expected time complexity of $\tilde{O}(n^3)$ in $\mathbb{R}^2$ and $\tilde{O}(n^5)$ in $\mathbb{R}^3$. As a byproduct, our framework also improves the integral bottleneck distance computation from $\tilde{O}(n^6)$ to $\tilde{O}(n^5)$ in 2D. Using similar techniques, the integral version in $\mathbb{R}^3$ can be computed in $\tilde{O}(n^7)$ time by maintaining matchings across all $O(n^6)$ events, though we leave the technical details for future work.

Our adaptation of the framework introduced by Bjerkevik and Kerber [1] reveals a broader principle: directional transforms that vary continuously over parameter spaces can often be optimized through event-driven algorithms and randomized candidate pruning. This suggests potential applications beyond the PHT to other parametric problems in computational topology, such as the Euler Characteristic Transform or persistence-based shape signatures.

From a practical perspective, the cubic complexity in 2D makes exact PHT comparison feasible for moderate-sized datasets, potentially enabling new applications in shape matching and retrieval where sampling-based approximations were previously necessary. However, significant challenges remain: extending to the Wasserstein distance would require fundamentally different techniques due to its non-combinatorial nature, and handling higher-dimensional ambient spaces faces steep complexity barriers.

References

1 Havard Bakke Bjerkevik and Michael Kerber. Asymptotic improvements on the exact matching distance for 2-parameter persistence. *Journal of Computational Geometry*, 14(1):309–342, December 2023. doi:10.20382/jocg.v14i1a12.

2 Sergio Cabello, Siu-Wing Cheng, Otfried Cheong, and Christian Knauer. Geometric Matching and Bottleneck Problems. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.31.

3 Mathieu Carrière, Frederic Chazal, Yuichi Ike, Theo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2786–2796. PMLR, 26–28 August 2020. URL: https://proceedings.mlr.press/v108/carriere20a.html.

4 David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. Vines and vineyards by updating persistence in linear time. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, SCG '06, pages 119–126, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1137856.1137877.

5 Justin Curry, Sayan Mukherjee, and Katharine Turner. How many directions determine a shape and other sufficiency results for two topological transforms. *Transactions of the American Mathematical Society, Series B*, 2018.

6 Tamal K. Dey and Cheng Xin. Computing Bottleneck Distance for 2-D Interval Decomposable Modules. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2018.32.

7 Tamal Krishna Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 1 edition, 2017. arXiv:1708.09359v2.

8 Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010.

9 Herbert Edelsbrunner and Teresa Heiss. Merge trees of periodic filtrations, 2024. doi:10.48550/arXiv.2408.16575.

10 Alon Efrat, Alon Itai, and Matthew J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, September 2001. doi:10.1007/s00453-001-0016-8.

11 Robert Ghrist, Rachel Levanger, and Huy Mai. Persistent homology and Euler integral transforms. *Journal of Applied and Computational Topology*, 2(1):55–60, October 2018. doi:10.1007/s41468-018-0017-1.

12 Chad Giusti, Eva Pastalkova, Carina Curto, and Vladimir Itskov. Clique topology reveals intrinsic geometric structure in neural correlations. *Proceedings of the National Academy of Sciences*, 112(44):13455–13460, 2015. doi:10.1073/pnas.1506407112.

13 Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, Volume 4 - 2021, 2021. doi:10.3389/frai.2021.681108.

14 Yasuaki Hiraoka, Takenobu Nakamura, Akihiko Hirata, Emerson G. Escolar, Kaname Matsue, and Yasumasa Nishiura. Hierarchical structures of amorphous solids characterized by persistent homology. *Proceedings of the National Academy of Sciences*, 113(26):7035–7040, 2016. doi:10.1073/pnas.1520877113.

15 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, December 1973. doi:10.1137/0202019.

16 Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *ACM J. Exp. Algorithmics*, 22, September 2017. doi:10.1145/3064175.

17 Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 2nd edition, 1998. Section 5.1.1.

18 Elizabeth Munch, Elena Xinyi Wang, and Carola Wenk. The kinetic hourglass data structure for computing the bottleneck distance of dynamic data, 2025. doi:10.48550/arXiv.2505.04048.

19 Michael W. Reimann, Max Nolte, Martina Scolamiero, Katharine Turner, Rodrigo Perin, Giuseppe Chindemi, Paweł Dłotko, Ran Levi, Kathryn Hess, and Henry Markram. Cliques of neurons bound into cavities provide a missing link between structure and function. *Frontiers in Computational Neuroscience*, Volume 11 - 2017, 2017. doi:10.3389/fncom.2017.00048.

20 Katharine Turner, Sayan Mukherjee, and Doug M. Boyer. Persistent homology transform for modeling shapes and surfaces. *Information and Inference: A Journal of the IMA*, 3(4):310–344, December 2014. doi:10.1093/imaiai/iau011.

Unavoidable Patterns and Plane Paths in Dense Topological Graphs

Balázs Keszegh
HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary
ELTE Eötvös Loránd University, Budapest, Hungary

Andrew Suk
Department of Mathematics, University of California at San Diego, La Jolla, CA, USA

Gábor Tardos
HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary

Ji Zeng
HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary

Abstract

Let $C_{s,t}$ be the complete bipartite geometric graph, with s and t vertices on two distinct parallel lines respectively, and all st straight-line edges drawn between them. In this paper, we show that every complete bipartite simple topological graph, with parts of size $2(k-1)^4+1$ and $2^{k^{5k}}$, contains a topological subgraph weakly isomorphic to $C_{k,k}$. As a corollary, every n-vertex simple topological graph not containing a plane path of length k has at most $O_k(n^{2-8/k^4})$ edges. When $k=3$, we obtain a stronger bound by showing that every n-vertex simple topological graph not containing a plane path of length 3 has at most $O(n^{4/3})$ edges. We also prove that x-monotone simple topological graphs not containing a plane path of length 3 have at most a linear number of edges.

2012 ACM Subject Classification Mathematics of computing → Extremal graph theory; Mathematics of computing → Graphs and surfaces

Keywords and phrases graph drawing, topological graph, bipartite geometric graph, forbidden subgraph, extremal graph, thrackle

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.63

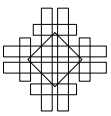
Related Version *Full version*: `https://arxiv.org/pdf/2512.04795` [16]

Funding This material is based upon work supported by the National Science Foundation under Grant No. DMS-1928930, while the authors were in residence at the Simons Laufer Mathematical Sciences Institute in Berkeley, California, during the 2025 Extremal Combinatorics Program.
Balázs Keszegh: Supported by the ERC Advanced Grant "ERMiD", no. 101054936 and by the EXCELLENCE-24 project no. 151504 Combinatorics and Geometry of the NRDI Fund.
Andrew Suk: Supported by NSF grant DMS-2246847.
Gábor Tardos: Supported by ERC Advanced Grants "GeoScape", no. 882971 and "ERMiD", no. 101054936.
Ji Zeng: Supported by ERC Advanced Grants "GeoScape", no. 882971 and "ERMiD", no. 101054936.

1 Introduction

A *topological graph* is a graph drawn in the plane such that its vertices are represented by points and its edges are represented by non-self-intersecting arcs connecting the corresponding points. No edge is allowed to pass through any point representing a vertex other than its endpoints. Tangencies between the edges are not allowed. That is, if two edges share an interior point, then they must properly cross at this point. A topological graph is called *plane* if there are no crossing edges. Given a topological graph G, we say that H is a *topological subgraph* of G if $V(H) \subset V(G)$ and $E(H) \subset E(G)$. We say that G and H are


© Balázs Keszegh, Andrew Suk, Gábor Tardos, and Ji Zeng;
licensed under Creative Commons License CC-BY 4.0



weakly isomorphic if there is an incidence preserving bijection between the vertices and edges of G and H such that two edges of G cross if and only if the corresponding edges in H cross as well. A topological graph is *simple* if every pair of its edges intersect at most once: at a common endpoint or at a proper crossing. Simple topological graphs are also known as *simple drawings*. If the edges of a topological graph are drawn with straight-line segments, then it is called *geometric*. We call a geometric graph *convex* if its vertices are in convex position.

In this paper, we are interested in finding large unavoidable patterns in dense simple topological graphs, and in particular, finding large plane paths. Let us emphasize here that a *path of length* k consists of $k+1$ distinct vertices and k edges. It is not hard to see that the simple condition here is necessary for plane paths, as one can easily draw K_n in the plane such that every pair of edges cross. Moreover, a construction due to Pach and Tóth [19] shows that there is a drawing of K_n in the plane such that every pair of edges crosses exactly once or twice. In 2003, Pach, Solymosi, and Tóth [20] showed that every complete n-vertex simple topological graph contains a topological subgraph on $\Omega(\log^{1/8}(n))$ vertices that is weakly isomorphic to either a complete convex geometric graph or a so-called complete twisted graph. This bound was later improved by Suk and Zeng [27] to $(\log n)^{1/4-o(1)}$. In 1998, Negami proved a bipartite analogue of this theorem. Let $C_{s,t}$ be a complete bipartite geometric graph with vertex sets U and V, where $|U| = s$ and $|V| = t$, such that the vertices in U lie on the y-axis and the vertices in V lie on the vertical line $x = 1$. (It is easy to see that $C_{s,t}$ is determined up to weak isomorphism independent of the exact placement of the vertices on these vertical lines.) In [18], Negami showed that for every $k > 1$, there is a minimum integer $n = n(k)$ such that every complete bipartite simple topological graph with n vertices in each part contains a topological subgraph weakly isomorphic to $C_{k,k}$. The proof in [18] is based on 4-uniform hypergraph Ramsey theory and no explicit bound for $n(k)$ is given. By applying more geometric arguments, we establish the following stronger result.

▶ **Theorem 1.** *Every complete bipartite simple topological graph with vertex sets U and V, where $|U| > 2(k-1)^4$ and $|V| \geq 2^{k^{5k}}$, contains a topological subgraph weakly isomorphic to $C_{k,k}$.*

We suspect that Theorem 1 still holds if $|V|$ is at least single exponential in a power of k, rather than double exponential in $\Omega(k \log k)$. On the other hand, our next result shows that the size of $U \cup V$ needs to be at least exponential in $k/2$.

▶ **Proposition 2.** *For every positive integer k there exists a complete simple topological graph on $2^{\lfloor k/2 \rfloor}$ vertices that does not contain a topological subgraph weakly isomorphic to $C_{k,k}$.*

Very recently, Balogh, Parada, and Salazar [5] extended Negami's result to complete multipartite simple topological graphs. Let us remark that our quantitative bound for the bipartite case, i.e. Theorem 1, is particularly interesting, because we can combine it with the celebrated Kővári-Sós-Turán theorem to conclude the following statement.

▶ **Theorem 3.** *Every n-vertex simple topological graph without a topological subgraph weakly isomorphic to $C_{k,k}$ has at most $O_k(n^{2-\epsilon(k)})$ edges with $\epsilon(k) = (2(k-1)^4 + 1)^{-1}$.*

It is an interesting open question to determine the optimal value of $\epsilon(k)$ in the theorem above.

An old conjecture due to Rafla [23] predicts that every complete n-vertex simple topological graph contains a plane Hamiltonian cycle for $n \geq 3$. This has been verified for small n and is open for $n \geq 10$ (see [1]). The much weaker conjecture of finding a linear size plane matching

in a complete n-vertex simple topological graph has been extensively studied [26, 3, 25]. The best known result for this problem is due to Aichholzer–García–Tejel–Vogtenhuber–Weinberger [3], who showed the existence of a plane matching of size $\Omega(\sqrt{n})$. Even less is known about plane paths. Recently, it is proved [3, 27] that every complete n-vertex simple topological graph contains a plane path of length $\Omega(\log n/\log\log n)$. Since one can easily find a plane path of length k in $C_{\lceil k/2\rceil,\lceil k/2\rceil}$, we can apply Theorem 3 to obtain the following corollary.

▶ **Corollary 4.** *Every n-vertex simple topological graph not containing a plane path of length k has at most $O_k(n^{2-8/k^4})$ edges.*

Next, we concentrate on the case of Corollary 4 when $k = 3$, that is, on simple topological graphs with no plane path of length 3. Such graph drawings can be regarded as a local variant of the so-called thrackles. A *thrackle* is a simple topological graph such that every pair of edges that do not share a vertex must cross. The famous thrackle conjecture of Conway states that a thrackle with n vertices has at most n edges, see [22] for a detailed history of the problem. For the special case of geometric graphs, the thrackle conjecture is proved by Erdős, and later, a short proof is given by Perles (see [22]). The proof of Perles also works if the drawing is outerplanar [7], i.e., if the points lie on the boundary of a disk and the edges lie inside this disk. The case that the edges are x-monotone is settled by Pach and Sterling [22]. In general, after a long series of improvements (Lovász–Pach–Szegedy [17] proved $2n$, Cairns–Nikolayevsky [6] proved $1.5n$, improved further by Fulek–Pach [11, 12] and Goddyn–Xu [13]), the current best upper bound on the number of edges of an n-vertex thrackle is $1.393(n-1)$ by Xu [30]. Apart from improvements on the upper bound, several papers consider variations of thrackles, in particular generalized thrackles [6], superthrackles [8], thrackles on surfaces [14], and thrackles of convex sets [4, 15].

We call a simple topological graph with no plane path of length 3 a *local thrackle*. Note that every thrackle is a local thrackle, but the two notions do differ. For example, consider the graph G on 6 vertices and 7 edges obtained from two disjoint copies of K_3 and an edge between them. Then G does not have a thrackle drawing (see, e.g., [29]), but G does have a local thrackle drawing (see full version for such a drawing [16]). However, our next result shows that the number of edges in a local thrackle with straight-line edges cannot be larger than n, just like in the case of thrackles.

▶ **Proposition 5.** *A local thrackle on n vertices and with straight-line segments as edges has at most n edges.*

Our proof in fact gives a characterization of the graphs that have a local thrackle drawing with segments. On the other hand, already with x-monotone edges one can draw a local thrackle with more than n edges, unlike in the case of thrackles.

▶ **Proposition 6.** *A local thrackle on n vertices and with x-monotone edges has at most $\frac{4}{3}n$ edges, while there exist local thrackles on n vertices with x-monotone edges and with $\frac{6}{5}n - O(1)$ edges.*

For the general case, we have a slightly better construction and an upper bound which improves the bound that comes from Corollary 4.

▶ **Proposition 7.** *A local thrackle on n vertices has at most $O(n^{4/3})$ edges, while there exist local thrackles on n vertices and with $2n - O(\sqrt{n})$ edges.*

Notice that the lower bound construction for the general case has more edges than the upper bound for the x-monotone case.

Our paper is organized as follows. In the next section, we prove Theorem 1 and Proposition 2. In Section 3, we establish our results on local thrackles, proving Propositions 5, 6, and 7. Finally, in Section 4, we give some concluding remarks. We systematically omit floors and ceilings whenever they are not crucial for the sake of clarity in our presentation. All logarithms are in base 2.

2 Unavoidable patterns in complete bipartite topological graphs

In this section, we prove Theorem 1. We will need the following combinatorial lemmas. Let G be an ordered graph with vertex set $U = \{u_1, u_2, \dots, u_n\}$. We say that G contains a *monotone path* of length k if there are $k+1$ indices $1 \le i_0 < i_1 < \cdots < i_k \le n$ such that $u_{i_j}u_{i_{j+1}} \in E(G)$ for all $0 \le i < k$. We say that G is *transitive* if for any $1 \le i_1 < i_2 < i_3 \le n$, the condition $u_{i_1}u_{i_2}, u_{i_2}u_{i_3} \in E(G)$ implies that $u_{i_1}u_{i_3} \in E(G)$. We will need the following three simple observations. The first is self evident, for the second we provide a simple proof to be self contained. The last one is a well known estimate of the multicolor Ramsey numbers.

▶ **Lemma 8.** *The vertices of a monotone path induce a clique in a transitive ordered graph.*

▶ **Lemma 9.** *Let t and m be positive integers. If G is a complete ordered graph on $n > tm^4$ vertices whose edges are colored with five colors, then either there exists an m-edge monochromatic monotone path in G in one of the first four colors, or there exists a monochromatic clique of size $t+1$ in the fifth color.*

Proof. Label each vertex v of G by the four-tuple (l_1, l_2, l_3, l_4), where l_i is the edge number of the longest monotone path in color i ending at v. If any of these paths have at least m edges we are done. If this is not the case, then all the labels are from $\{0, \dots, m-1\}^4$, so by the pigeonhole principle, some $t+1$ vertices must share the same label. We claim that these vertices induce a monochromatic clique in the fifth color. Indeed, assume the vertex u precedes vertex v in the vertex order and the edge uv receives color i. Consider the longest monotone path in color i ending at u. Extending this path with the uv edge we get a longer monotone path in color i ending at v showing that u and v cannot share the same label unless $i = 5$. This proves the lemma. ◀

Let $r(k; r)$ be the minimum integer n such that every r-coloring of the edges of the complete graph on n vertices contains a monochromatic clique on k vertices. The existence of $r(k; r)$ follows by the famous theorem of Ramsey [24], and following the arguments of Erdős–Szekeres [9], we have

▶ **Lemma 10.** $r(k; r) < r^{rk}$.

Proof of Theorem 1. Let G be a complete bipartite simple topological graph whose vertex set is $U \cup V$, where $|U| = m > 2(k-1)^4$ and $|V| = n \ge 2^{k^{5k}}$. We assume without loss of generality that $m = 2(k-1)^4 + 1$ (otherwise we simply ignore the extra vertices in U). There is nothing to prove when $k = 1$. When $k = 2$, since $K_{3,3}$ is not planar, we can find a pair of crossing edges u_1v_1 and u_2v_2 in G, and as G is simple, the induced topological subgraph on u_1, u_2, v_1, v_2 is weakly isomorphic to $C_{2,2}$. Therefore, we assume $k \ge 3$ for the rest of this proof.

By arbitrarily ordering the elements in U and V, we have $U = \{u_1, \dots, u_m\}$, and $V = \{v_1, \dots, v_n\}$. For each 4-tuple $u_i, u_j \in U$ and $v_s, v_t \in V$, where $i < j$ and $s < t$, let us consider the drawing of $K_{2,2}$ induced on these vertices. Note that as we have a simple topological graph, only non-adjacent edges can cross, that is, u_iv_s may cross u_jv_t or u_iv_t may

cross u_jv_s. If both of these pairs of edges cross, then we obtain a thrackle drawing of C_4 and a small case-analysis shows that such a drawing does not exist. So we have three possibilities, either the first pair cross, or the second pair, or it is a plane drawing. We further refine this classification according to the orientation of the vertices around the crossing point obtaining the following five *order types*. When speaking about the order of the vertices around the crossing point we mean the cyclic order of the edge-segments going to these vertices.

1. The 4-tuple $\{u_i, u_j, v_s, v_t\}$ is of *type 1* if edge u_iv_t crosses edge u_jv_s, and moreover, vertices (u_i, v_s, v_t, u_j) appear in clockwise order from the crossing point of u_iv_t and u_jv_s.
2. The 4-tuple $\{u_i, u_j, v_s, v_t\}$ is of *type 2* if edge u_iv_t crosses edge u_jv_s, and moreover, vertices (u_i, u_j, v_t, v_s) appear in clockwise order from the crossing point of u_iv_t and u_jv_s.
3. The 4-tuple $\{u_i, u_j, v_s, v_t\}$ is of *type 3* if edge u_iv_s crosses edge u_jv_t, and moreover, vertices (u_i, v_t, v_s, u_j) appear in clockwise order from the crossing point of u_iv_s and u_jv_t.
4. The 4-tuple $\{u_i, u_j, v_s, v_t\}$ is of *type 4* if edge u_iv_s crosses edge u_jv_t, and moreover, vertices (u_i, u_j, v_s, v_t) appear in clockwise order from the crossing point of u_iv_s and u_jv_t.
5. The 4-tuple $\{u_i, u_j, v_s, v_t\}$ is of *type 5* if it induces a plane drawing of $K_{2,2}$.

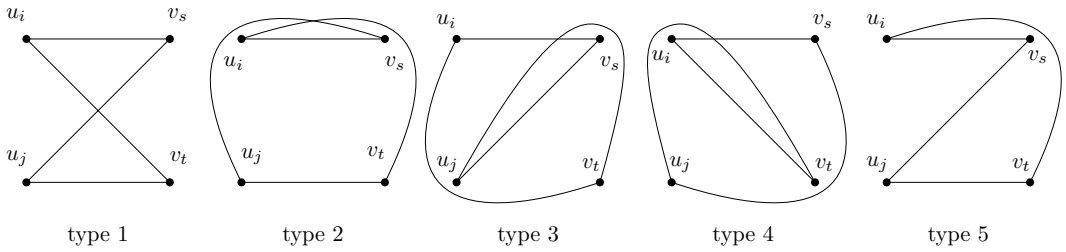


Figure 1 A drawing of each of the five types of C_4's.

See Figure 1. For $v_s, v_t \in V$ and $w \in \{1, 2, 3, 4, 5\}$, let $G^{(w)}_{s,t}$ be the ordered graph whose vertex set is U, and edges are pairs of vertices $\{u_i, u_j\}$ such that the 4-tuple $\{u_i, u_j, v_s, v_t\}$ is of type w.

We claim that the ordered graph $G^{(w)}_{s,t}$ is transitive for $w \in \{1, 2, 3, 4\}$, that is, with respect to the first four order types. We verify the statement for $G^{(1)}_{s,t}$. For $i < j < \ell$, suppose that $\{u_i, u_j\}$ and $\{u_j, u_\ell\}$ are edges in $G^{(1)}_{s,t}$. Consider the drawing of $K_{2,2}$ induced by the 4-tuple $\{u_i, u_j, v_s, v_t\}$, specifically the "triangle" formed by u_j, v_t and the crossing point between u_jv_s and u_iv_t. Since (u_j, v_s, v_t, u_ℓ) must appear in clockwise order from the crossing point of u_jv_t and $u_\ell v_s$, we must exit the "triangle" at this crossing point as we travel along $u_\ell v_s$ from v_s to u_ℓ. This implies that $u_\ell v_s$ crosses both u_jv_t and u_iv_t. Further, u_ℓ, v_s, and v_t must appear in the same cyclic order from the crossing point of u_jv_t and $u_\ell v_s$ as well as from the crossing point of u_iv_t and $u_\ell v_s$. The former is clockwise order from the assumption $\{u_j, u_\ell\} \in E(G^{(1)}_{s,t})$, so the latter also clockwise implying $\{u_i, u_\ell\} \in E(G^{(1)}_{s,t})$ as claimed. See Figure 2. The transitivity property of $G^{(2)}_{s,t}, G^{(3)}_{s,t}$, and $G^{(4)}_{s,t}$ can be argued similarly, or be reduced to the case of $G^{(1)}_{s,t}$ by reversing the vertex orders on U or V. For example, a type 3 quadruple $\{u_i, u_j, v_s, v_t\}$ would have been type 1 if we had ordered the vertices in U the same but in V reversely.

Since $|U| > 2(k-1)^4$, by combining Lemmas 8, 9, and the fact that $G^{(w)}_{s,t}$ is transitive when $w \in \{1, 2, 3, 4\}$, we have the following observation.

▷ Claim 11. For each pair $v_s, v_t \in V$, there is a subset $S \subset U$ such that either

1. $|S| = k$, and S induces a clique in $G^{(w)}_{s,t}$ for some $w \in \{1, 2, 3, 4\}$, or
2. $|S| = 3$, and S induces a clique (i.e., a triangle) in $G^{(5)}_{s,t}$.

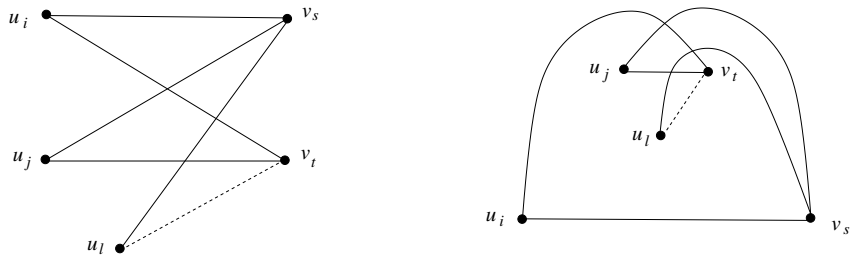


Figure 2 Proof of transitivity of $G^{(1)}_{s,t}$.

This claim implies that we can label the edge v_sv_t of the complete graph on V with a pair (S, i), where $i \in \{1, \dots, 5\}$ and $S \subset U$ is a clique in $G^{(i)}_{s,t}$ of size 3 if $i = 5$ or of size k otherwise. In case there are more possible labels we choose arbitrarily. For $k \geq 7$, using $|U| = 2(k-1)^4 + 1 \leq 2k^4$, the number of distinct labels we use on the edges is at most

$$r(k) := 4\binom{|U|}{k} + \binom{|U|}{3} \leq 5\binom{2k^4}{k} \leq 5(2ek^3)^k \leq k^{4k}.$$

And we can further estimate using $k \geq 7$

$$r(k)^{r(k)\cdot k} < 2^{k^{5k}}.$$

For $k = 3, 4, 5, 6$, the upper bound for $r(k)$ is too crude, but we can verify the previous inequality by direct computation.

Applying Lemma 10 together with $|V| \geq 2^{k^{5k}}$, there is a subset $T \subset V$ of size k such that every pair in T has the same label (S, i). Let us consider the topological subgraph G' induced on $S \cup T$. The proof now falls into the following cases.

Case 1. Suppose $|S| = k$ and $i \in \{1, 2, 3, 4\}$. We can assume that $i = 1$, since a symmetric argument would follow otherwise. Then for $u_i, u_j \in S$ and $v_s, v_t \in T$, where $i < j$ and $s < t$, $\{u_i, u_j, v_s, v_t\}$ is of type 1, and therefore, u_iv_t crosses u_jv_s. Hence, G' is weakly isomorphic to $C_{k,k}$.

Case 2. Suppose $|S| = 3$ and $i = 5$. Then G' is a plane drawing of $K_{3,k}$. This is impossible as $k \geq 3$. ◀

2.1 Construction: Proof of Proposition 2

Here we show that one needs exponentially many vertices in a simple topological graph – even in a simple complete topological graph – to ensure the presence of a subgraph weakly isomorphic to $C_{k,k}$. Note however, that this lower bound does not match the doubly exponential bound in Theorem 1.

Fix $k \geq 3$. We construct a large random simple complete topological graph as follows, and show that with positive probability it has no subgraph weakly isomorphic to $C_{k,k}$. Set $n = 2^{\lfloor k/2 \rfloor}$. Let U be a set of n points placed on the x-axis. This will be the vertex set of the graph we construct. For each pair $u, v \in U$, draw an edge connecting u and v as a half-circle in the upper or lower half of the plane uniformly at random and independently for each pair (u, v). Clearly, this will result in a simple drawing of K_n in the plane.

Consider four vertices $u_1, u_2, u_3, u_4 \in U$ that appear in this order on the x axis. These vertices induce a drawing of K_4 in which only the edges u_1u_3 and u_2u_4 cross if and only if they are drawn in the same half plane. If these two edges are drawn in distinct half planes we obtain a plane drawing of K_4.

Let $S, T \subseteq U$ be disjoint sets of size k and let us calculate the probability that the edges between S and T form a subgraph weakly isomorphic to $C_{k,k}$. First note that $C_{k,k}$ has no subgraph that is a plane drawing of C_4, so for the weak isomorphism we need that for every $u \neq u' \in S$ and $v \neq v' \in T$ some two edges between $\{u, u'\}$ and $\{v, v'\}$ cross. This implies that the order on x axis cannot alternate between the vertices in S and the vertices of T, because in that case the subgraph is a plane drawing with probability one. This implies that either S can be split into two sets S_1 and S_2, so that all vertices in S_1 are to the left of the vertices of T and all vertices in S_2 are to the right of the vertices in T, or T can be split in two sets, one to the left and one to the right of S. We call such pairs *separated*.

Now let S and T be two separated k-element set of vertices. We may assume that S splits into S_1 and S_2 as above and S_1 is not empty (otherwise, switch the roles of S and T). Let v_0 and v_1 be the leftmost and rightmost vertices in T, respectively, while u_1 is the rightmost vertex in S_1 and u_0 is the leftmost vertex in S_2, or in case S_2 is empty, then u_0 is the leftmost vertex in S_1. We claim that the subgraph formed by the edges between S and T is weakly isomorphic to $C_{k,k}$ if and only if all of its edges, with the possible exception of u_1v_0 and u_0v_1, are drawn in the same (upper or lower) half plane. The if part of this equivalence is easy to check (and strictly speaking, not even needed for our argument). For the only if part assume without loss of generality that u_1v_1 is drawn in the upper half plane. Now consider two other vertices: $u_1 \neq u \in S$ and $v_1 \neq v \in T$. If uv is drawn in the lower half plane, then the 4-cycle u_1v_1uv is a plane drawing, so we do not have the desired weak isomorphism. So the weak isomorphism implies that all of these edges are drawn in the upper half plane including the edge u_0v_0. But then a similar argument shows that all the edges uv with $u_0 \neq u \in S$ and $v_0 \neq v \in T$ must also be drawn in the upper half plane. This covers all the edges mentioned in the claim.

As a result, the probability that the edges between the separated sets S and T form a subgraph weakly isomorphic to $C_{k,k}$ is exactly 2^{-k^2+3} and therefore the expected number of subgraphs weakly isomorphic to $C_{k,k}$ is exactly

$$k\binom{n}{2k}2^{-k^2+3}.$$

By our choice of $n = 2^{\lfloor k/2 \rfloor}$ and $k \geq 3$, this is less than 1, so the random drawing of K_n contains no subgraph weakly isomorphic to $C_{k,k}$ with positive probability.

3 Local thrackles

We say that a graph G is *thrackleable* (*local thrackleable*) if G has a thrackle drawing (local thrackle drawing). Likewise, we say that G is *geometric local thrackleable* if there is a drawing of G with straight-line edges that is a local thrackle.

Let us first consider the case when edges are drawn as straight-line segments. In this case, we can even classify all geometric local thrackleable graphs following the arguments of Woodall [29]. We prove Proposition 13 below which implies Proposition 5. A *caterpillar* is a tree in which each vertex is adjacent to at most two non-leaf vertices. A *spiked cycle* is a connected graph with one cycle such that every vertex not on this cycle has degree one. The *length* of a spiked cycle is the length of its unique subgraph that is a cycle. Let us first recall Woodall's characterization of geometric thrackles [29] using these definitions:

▶ **Theorem 12.** *A graph can be drawn as a thrackle with straight-line segments as edges if and only if either it is a union of disjoint caterpillars, or it is a spiked cycle of odd length with some additional isolated vertices.*

Our characterization of geometric local thrackles is the following:

▶ **Proposition 13.** *A graph can be drawn as a local thrackle with straight-line segments as edges if and only if it is the disjoint union of caterpillars, spiked cycles of odd length and spiked cycles of even length at least eight.*

Proof. First, we argue that if a graph G contains a vertex v with at least three non-leaf neighbors, then G is not geometric local thrackleable. See Figure 3 for two such graphs with G_1 already covered in [29]. Indeed, if v has three non-leaf neighbors, then in any straight line drawing, one of them "splits" the neighborhood of v, that is, v has neighbors in both sides of the line of the edge vw for some non-leaf neighbor w. But as w is no leaf it has another neighbor $w' \neq v$ and in any local thrackle drawing ww' intersects (crosses or has a common endpoint with) every edge incident to v, which is only possible if w does not split the neighborhood, a contradiction.

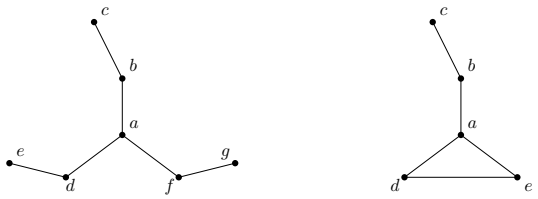


Figure 3 Two forbidden subgraphs G_1 (left) and G_2 (right) for geometric local thrackles.

Let us remove all the leaves of a geometric local thrackleable graph G. The observation above implies that the remaining graph has maximum degree at most 2, so it is the disjoint union of paths and cycles. This implies that G is the disjoint union of caterpillars and spiked cycles. To finish the only if part of the proposition we are left to show that C_4 and C_6 are not geometric local thrackleable. It is easy to see that if C_4 is drawn as a local thrackle, then this drawing must be a thrackle. The same holds true for C_6 as we shall prove in Lemma 15 (although for straight line drawings the case analysis could be shortened). On the other hand, it follows from Theorem 12 that even cycles are not geometric thrackleable (C_4 is not even thrackleable), hence C_4 and C_6 are not geometric local thrackleable.

For the if part of proposition, notice that the components of G can be drawn pairwise disjointly, so it is enough to prove that caterpillars, odd spiked cycles and even spiked cycles of length at least 8 are geometric local thrackleable. The first two cases are already handled by Theorem 12 (as those graphs are geometric thrackleable), so here we concentrate on even spiked cycles. Let $n \geq 8$ be even. We first place vertices $v_1, v_2, \ldots, v_{n-3}$ on the circle according to the following clockwise order: $v_1, v_3, v_5, \ldots, v_{n-3}$, $v_2, v_4, \ldots, v_{n-4}$. Then we put v_{n-2} on the clockwise arc v_1v_3, v_{n-1} on the clockwise arc v_2v_4, and v_n on the clockwise arc $v_{n-5}v_{n-3}$. Observe that we obtain a geometric local threackle drawing of C_n by connecting v_iv_{i+1} using straight-line segments. See Figure 4 for an illustration. Finally, we can choose an interval on the cycle for each v_i such that if we place the leaf neighbors of v_i on this intervals the resulting drawing is still a local thrackle. ◀

We continue with the case when edges are drawn x-monotone.

Proof of Proposition 6. For the lower bound we give a construction with $6/5n - O(1)$ edges. See Figure 5. First, we draw a path as seen on the top of Figure 5 as a local thrackle. Then we make a translated copy of this path as seen on the middle of Figure 5, this is clearly still a local thrackle. Finally, we add further paths of length 2, one of them is drawn blue on the

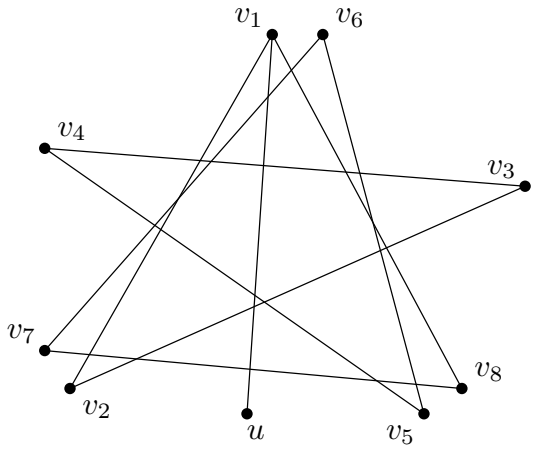


Figure 4 C_8 can be drawn as a linear local thrackle, and with a spike u attached to v_1.

middle of Figure 5, and moreover, one can check that adding this one path maintained that the drawing is a local thrackle. All such blue paths are drawn on the bottom of Figure 5, where it is not hard to check that the drawing gives a simple topological graph and also it is still a local thrackle (the blue 2-paths are far enough in the graph, thus no planar 3-path can be created when adding all blue 2-paths, if adding one did not create a planar 3-path), finishing the construction. Note that we connected using a 2-path every second vertex of one path with the corresponding vertex of the second path, thus when extending the construction with 2 vertices on each path plus a vertex to add the 2-path joining, we added 5 vertices and 6 edges. Thus, the number of edges is $6/5n - O(1)$ (the constant error comes from the fact that close to the two ends of the two paths some edges are missing).

For the upper bound, let G be a local thrackle with x-monotone edges. We may assume that every vertex has degree at least one. We distinguish three types of vertices. A vertex that has no edges going to the left (resp. right) is called a *left-vertex* (resp. *right-vertex*). A vertex that has edges in both directions is called a *middle-vertex*. Assume there are l left-vertices, r right-vertices and m middle-vertices, $l + r + m = n$.

At each left-vertex, delete the incident edge that leaves highest at this vertex. Delete at each right-vertex the incident edge that leaves lowest at this vertex. We have deleted at most $l + r$ edges. Notice that if there is a remaining edge between a left-vertex and a right-vertex, then together with the 2 deleted edges incident to these vertices, we get a plane 3-path, a contradiction. Therefore, no edge is left between a left- and a right-vertex. Similarly, if a middle-vertex is connected to two left-vertices in the remaining graph, then adding the deleted edge of the left-vertex on the higher of these two edges, we obtain a plane 3-path. The contradiction shows that every middle vertex is connected to at most one left-vertex in the remaining graph.

Next, we claim that the remaining edges connecting the left-vertices and middle-vertices form a matching. After the observation above it is enough to show that any left-vertex has at most one middle-vertex neighbor. Indeed, otherwise take two edges from a left-vertex to middle-vertices and two more edges going right from these middle-vertices. It is easy to see that one of the 3-paths in the obtained 4-path or 4-cycle must be plane, a contradiction. Therefore, the number of remaining edges connecting left- and middle vertices is at most $\min(l, m)$.

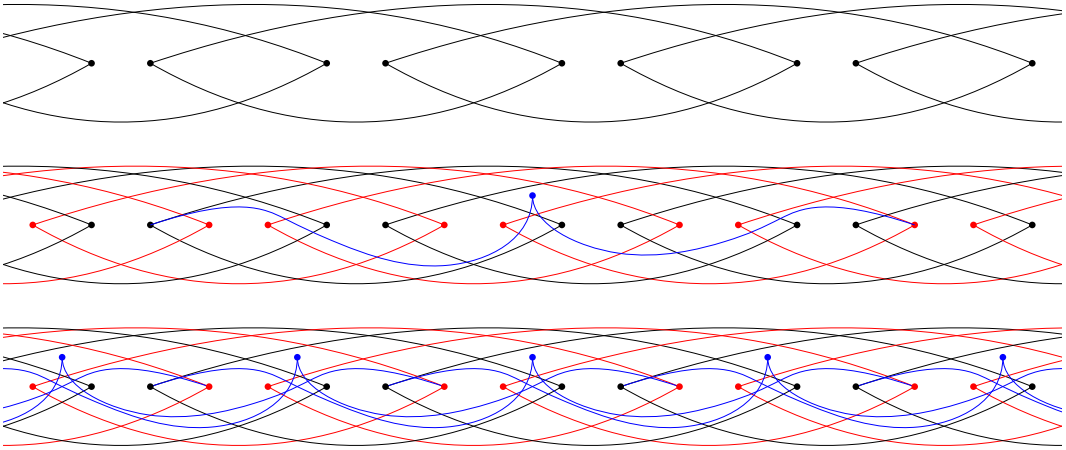


Figure 5 Two paths (black and red) joined by additional 2-paths (blue) drawn as a local thrackle with x-monotone edges.

Similarly, there are at most $\min(r, m)$ edges between the right-vertices and middle-vertices left in the graph. No edge connects two middle-vertices because that would lead to an x-monotone (therefore plane) 3-path. Putting everything together, our original graph has at most $l + r + \min(l, m) + \min(r, m)$ edges. Assume without loss of generality that $r \leq l$. Then this is at most $l + r + m + \min(l, r, m) = n + \min(l, r, m) \leq n + n/3 = 4n/3$. ◀

For the case of general local thrackles we need the following statements. Let Θ_3 denote a graph consisting of two vertices connected by three internally-disjoint paths of length three. That is, Θ_3 is a graph with 8 vertices and 9 edges.

▶ **Theorem 14** ([10]). *A graph that does not contain Θ_3 as a subgraph has at most $O(n^{4/3})$ edges.*

▶ **Lemma 15.** *Any local thrackle drawing of C_6 is also a thrackle drawing of C_6.*

Proof. Fulek and Pach [11] invented a computational approach to check whether a given graph is thrackleable. In fact, as we shall describe, their method is more general and can check whether any crossing pattern can be realized as a simple drawing or not. Let $G = (V, E)$ be an oriented graph. (We need the orientation for reference purposes only.) Suppose we are given an ordered list $\ell(e)$ consisting of elements in $E \setminus e$ for each $e \in E$, with the property that $f \in \ell(e)$ if and only if $e \in \ell(f)$, then we can construct an auxiliary graph H through the following procedure:

1. (Planarization) For every $e \in E$ directed from u to v, we replace the edge $e = \{u, v\}$ by a path from u to v, through the vertices named as $\{e, f\}$ with $f \in \ell(e)$ following the order of $\ell(e)$. Here, we introduce new vertices $\{e, f\}$ if necessary.
2. (Subdivision) For each edge between two new vertices created in the planarization step, we replace it with a path of length two, by inserting an additional vertex.
3. (Bracing) For each new vertex $v = \{e, f\}$ created in the planarization step, there will be exactly four vertices a, b, c, d adjacent to v (they may be original vertices in V or new vertices introduced in subdivision step), where a, b are adjacent to v along segments of of e and c, d are adjacent to v along segments of f. We create the edges $\{a, c\}$, $\{a, d\}$, $\{b, c\}$, and $\{b, d\}$.

We remark that here $\{e, f\}$ is unordered, so it will be considered in the planarization steps for e and f respectively, which will give it exactly four neighbors.

▷ Claim 16. The following two statements are equivalent: (i) There exists a drawing of G where the crossings on each edge e, along its direction, are exactly those between e and f with $f \in \ell(e)$ in this order. (ii) The graph H constructed above according to G and ℓ is planar.

We give a brief explanation to this claim. Suppose G has such a drawing, then we can turn it into a plane drawing by turning each crossing point into a new vertex (planarization). Next, we can subdivide each edge between the crossing points once (subdivision). Finally, we connect the neighbors of each crossing point in their cyclic order (bracing). This can be done easily such that the resulting drawing is still plane, implying that H is planar; On the other hand, suppose we have a plane drawing of H, we can draw any edge e of G, directed from u to v, along the path from u to v in H created in planarization. The vertices created in the subdivision step serve as a guide on which path to follow at a crossing point. The bracing structures will enforce the edges drawn to cross properly without tangencies. This will give a drawing of G as wanted.

Following this approach, we used a computer program to verify Lemma 15. To prove this lemma, it suffices for us to take $G = C_6$, direct its edges arbitrarily, create every possible crossing list ℓ that obeys the local thrackle condition, construct H and test for its planarity. If all such H except for those corresponding to thrackles are not planar, then we can conclude the proof. A source code for such a computer program is given in the full version of this paper [16]. ◀

▶ **Theorem 17** (Theorem 5.1 in [17]). *A thrackleable graph cannot contain Θ_3 as a subgraph.*

We show the following strengthening of the above theorem:

▷ Claim 18. A local thrackleable graph cannot contain Θ_3 as a subgraph.

Proof. Assume on the contrary, then Θ_3 can be drawn as a local thrackle. By Theorem 17 this is not a thrackle, therefore there are two independent edges that do not cross. The only way this can happen is that there is a subgraph C_6 drawn such that a pair of opposite edges do not cross. This is a local thrackle drawing of a C_6 which is not a thrackle drawing, contradicting Lemma 15, finishing the proof. ◁

▷ Claim 19. The 1-subdivision of the complete graph $K_{n,n}$ is local thrackleable.

Proof. First we show that a 1-subdivision of a star S_n is local thrackleable. For that see Figure 6. Note that outside of an area that can be made arbitrarily small (the inside of the dotted circle on the figure), the edges incident to the center can be drawn close to each other and the edges incident to the leaves can be drawn parallel. That is, every 2-path starting at v_1 goes very close to a given 2-path (say the one joining v_1 to w_1) and only in a small vicinity of the middle vertex of this given 2-path do these 2-paths intersect each other. Note that the cyclic order of the paths around v_i is reversed by the time they reach the w_i's.

We have a drawing of the 1-subdivision of the star S_n with center v_1 and leaves $W = \{w_1, \dots, w_n\}$. Now we draw n copies of this graph so that the leaves are identified to get the drawing of the graph G which is the 1-subdivision of $K_{n,n}$. The way we do it is that we put $n-1$ more vertices, $v_2, \dots, v_n$, close to v_1 and draw the new edges in the following way. For each w_i, we had one 2-path going to v_1 and we need $n-1$ more 2-paths going to $v_2, \dots, v_n$. We draw them the same way as we created the original star, just now w_i is the center and the 2-path from w_i to v_1 is the 2-path close to which all the other 2-paths go (and they only intersect in a small vicinity of the middle vertex of the 2-path from w_i to v_1). We denote $V = \{v_1, \dots v_n\}$.

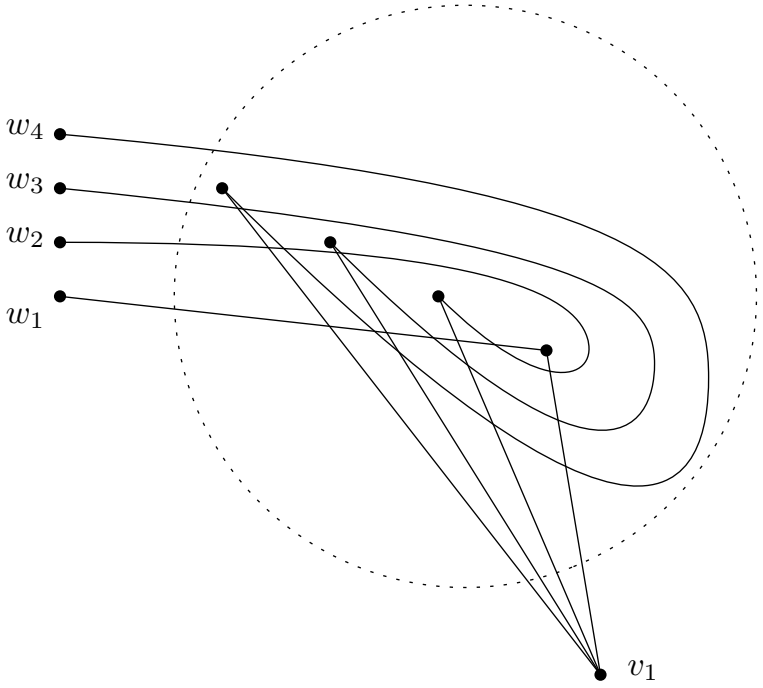


Figure 6 The 1-subdivision of a star drawn as a local thrackle.

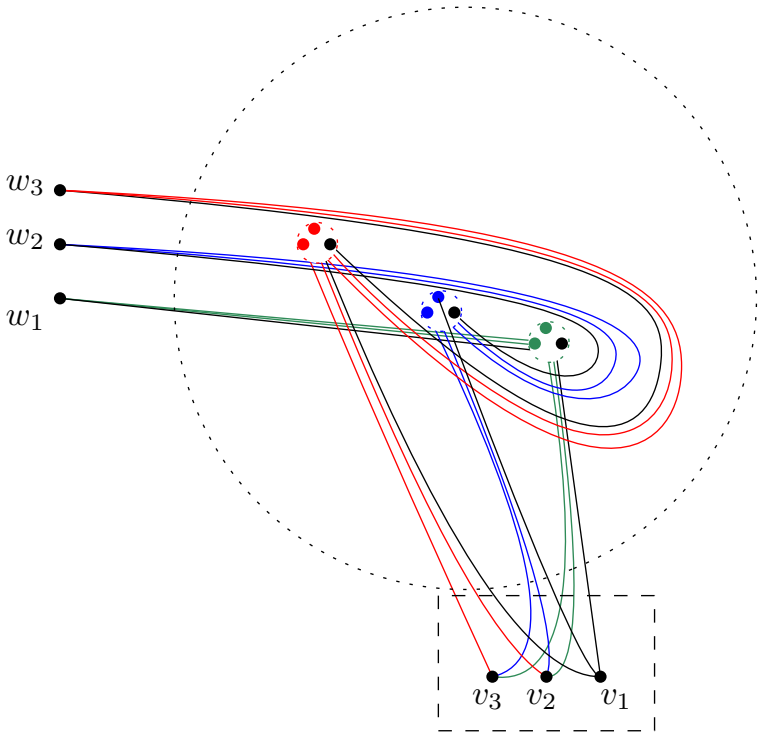


Figure 7 Subdivided $K_{3,3}$ drawn as a local thrackle. The intersections among the edges of the subdivided stars centered at w_i are hidden inside the dotted circles, each of them hiding a drawing that looks like the drawing on Figure 6 (with w_i and V on this figure playing the role of v_1 and W on Figure 6, respectively).

We draw the edges in the vicinity of v_1 such that they intersect pairwise at most once and two edges with a common endpoint v_i do not cross. See Figure 7, the dashed rectangle is the vicinity of v_1 where such crossings happen.

Finally, we are left to prove that this is a local thrackle. It is easy to check we have a simple drawing. It remains to check that no 3-edge path is planar. For this, notice that any 3-path contains exactly one vertex from V and W respectively. This implies that the 3 edges of any 3-path belong to a 1-subdivision of a star that was drawn as a local thrackle. For v_1 this follows from our drawing of S_n, for other v_i it follows from the fact that these edges go close to the respective edge from v_1, and for w_i it again follows from the fact that at w_i we drew the edges in distance two as a local thrackle. ◁

Proof of Proposition 7. The upper bound directly follows from Theorem 14 and Claim 18. The lower bound follows from Claim 19 by observing that in a 1-subdivision of $K_{n,n}$ we have $n^2 + 2n$ vertices and $2n^2$ edges. ◀

Note that Claim 19 implies that the 3-subdivision of K_n is local thrackleable, which implies that the 3-subdivision of any graph is local thrackleable. Therefore arguments that rely on forbidden subdivided copies of some graph (e.g., planarity arguments) won't work when trying to improve the upper bound in Proposition 7.

In the full version of this paper [16] we give further examples of local thrackles.

4 Related problems and concluding remarks

1. A *dual* version of Proposition 6 was obtained by Pach, Pinchasi, Tardos, and Tóth in [21], who proved that every n-vertex x-monotone simple topological graph with no self-intersecting path of length 3 has at most $O(n \log n)$ edges. Moreover, they showed that this bound is asymptotically tight. For fixed $k \geq 3$, a construction due to Tardos [28] shows that there are geometric graphs with no self-intersecting path of length k and with a superlinear number of edges. Pach, Pinchasi, Tardos, and Tóth in [21] also showed that every n-vertex topological graph (not necessarily simple) with no self-intersecting path of length 3 has at most $O(n^{3/2})$ edges. In the full version of the paper [16], we give a short proof of the following variant.

▶ **Theorem 20.** *Every n-vertex simple topological graph G with no self-intersecting path of length 4 has at most $O(n \log^2 n)$ edges.*

2. Two topological graphs G and H are *isomorphic* if there is a homeomorphism of the sphere that transforms G to H, where this sphere is the one-point compactification of the plane we draw the graphs on. The proof of Theorem 1 finds more than a weak-isomorphic copy of $C_{k,k}$ any simple drawing of a large enough complete graph. The drawing of $K_{k,k}$ we find as a subgraph agrees with $C_{k,k}$ in its *extended rotation system*, that is, the cyclic orders of edges emanating from all vertices and crossings. According to Theorem 1 in the recent paper [2], this determines H up to triangle flips followed by an isomorphism. Here, a *triangle flip* refers to the operation of moving one edge of a triangular cell formed by three pairwise crossing edges over the opposite crossing of the two other edges.

3. The following 4-uniform hypergraph Ramsey problem was also studied independently by Negami [18] and Mathias Schacht (private communication). Let $S_{a,b} = (U, V, E)$ be the 4-uniform hypergraph with the vertex set the disjoint union of U and V, where $|U| = a$ and $|V| = b$, such that $E(S_{a,b}) = \{\{x, y, z, w\} : x \neq y, z \neq w, x, y \in U, \text{ and } z, w \in V\}$. What is the minimum integer $r(S_{n,n}) = N$ such that for any red/blue coloring of the edges of

$S_{N,N}$, there is a monochromatic copy of $S_{n,n}$? Following the arguments given in Section 2, one can show that $r(S_{n,n}) < 2^{2^{O(n^2)}}$, while a standard probabilistic argument shows that $r(S_{n,n}) > 2^{\Omega(n^3)}$.

References

1 Bernardo M Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Thomas Hackl, Jürgen Pammer, Alexander Pilz, Pedro Ramos, Gelasio Salazar, and Birgit Vogtenhuber. All good drawings of small complete graphs. In *Proc. 31st European Workshop on Computational Geometry (EuroCG)*, pages 57–60, 2015.

2 Oswin Aichholzer, Man-Kwun Chiu, Hung P Hoang, Michael Hoffmann, Jan Kynčl, Yannic Maus, Birgit Vogtenhuber, and Alexandra Weinberger. Drawings of complete multipartite graphs up to triangle flips. *Journal of Computational Geometry*, 15(2):124–173, 2024.

3 Oswin Aichholzer, Alfredo García, Javier Tejel, Birgit Vogtenhuber, and Alexandra Weinberger. Twisted ways to find plane structures in simple drawings of complete graphs. *Discrete & Computational Geometry*, 71(1):40–66, 2024. doi:10.1007/S00454-023-00610-0.

4 Megumi Asada, Ryan Chen, Florian Frick, Frederick Huang, Maxwell Polevy, David Stoner, Ling Hei Tsang, and Zoe Wellner. On Reay's relaxed Tverberg conjecture and generalizations of Conway's thrackle conjecture. *The Electronic Journal of Combinatorics*, 25(3):#P3.16, 2018.

5 József Balogh, Irene Parada, and Gelasio Salazar. The unavoidable drawings of complete multipartite graphs. *arXiv preprint*, 2025. arXiv:2509.20625.

6 Grant Cairns and Yury Nikolayevsky. Bounds for generalized thrackles. *Discrete & Computational Geometry*, 23:191–206, 2000. doi:10.1007/PL00009495.

7 Grant Cairns and Yury Nikolayevsky. Outerplanar thrackles. *Graphs and Combinatorics*, 28:85–96, 2012. doi:10.1007/S00373-010-1010-1.

8 Hooman Dehkordi and Graham Farr. Thrackles, superthrackles and the Hanani-Tutte theorem. *Journal of Graph Algorithms and Applications*, 28(1):95–127, May 2024. doi:10.7155/JGAA.V28I1.2930.

9 Pál Erdős and György Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

10 Ralph J Faudree and Miklós Simonovits. On a class of degenerate extremal graph problems. *Combinatorica*, 3:83–93, 1983. doi:10.1007/BF02579343.

11 Radoslav Fulek and János Pach. A computational approach to Conway's thrackle conjecture. *Computational Geometry*, 44(6-7):345–355, 2011. doi:10.1016/J.COMGEO.2011.02.001.

12 Radoslav Fulek and János Pach. Thrackles: An improved upper bound. *Discrete Applied Mathematics*, 259:226–231, 2019. doi:10.1016/J.DAM.2018.12.025.

13 Luis Goddyn and Yian Xu. On the bounds of Conway's thrackles. *Discrete & Computational Geometry*, 58:410–416, 2017. doi:10.1007/S00454-017-9877-8.

14 César Hernández-Vélez, Jan Kynčl, and Gelasio Salazar. Thrackles on nonplanar surfaces. *arXiv preprint*, 2025. arXiv:2506.11808.

15 Balázs Keszegh and Dániel Simon. Convex hull thrackles. *Discrete Mathematics*, 349(3):114840, 2026. doi:10.1016/J.DISC.2025.114840.

16 Balázs Keszegh, Andrew Suk, Gábor Tardos, and Ji Zeng. Unavoidable patterns and plane paths in dense topological graphs. *arXiv preprint*, 2025. arXiv:2512.04795.

17 László Lovász, János Pach, and Mario Szegedy. On Conway's Thrackle Conjecture. *Discrete & Computational Geometry*, 18:369–376, 1997. doi:10.1007/PL00009322.

18 Seiya Negami. Ramsey-type theorems for spatial graphs and good drawings. *Journal of Combinatorial Theory, Series B*, 72:53–62, 1998. doi:10.1006/JCTB.1997.1783.

19 J. Pach and G. Tóth. Disjoint edges in topological graphs. *Journal of Combinatorics*, 1:335–344, 2010.

20 János Pach, József Solymosi, and Géza Tóth. Unavoidable Configurations in Complete Topological Graphs. *Discrete & Computational Geometry*, 30:311–320, 2003. doi:10.1007/S00454-003-0012-9.

21 János Pach, Rom Pinchasi, Gábor Tardos, and Géza Tóth. Geometric graphs with no self-intersecting path of length three. *European Journal of Combinatorics*, 25(6):793–811, 2004. doi:10.1016/J.EJC.2003.09.019.

22 János Pach and Ethan Sterling. Conway's conjecture for monotone thrackles. *The American Mathematical Monthly*, 118(6):544–548, 2011. doi:10.4169/AMER.MATH.MONTHLY.118.06.544.

23 Nabil H. Rafla. The good drawings of the complete graph. *PhD Thesis, McGill University, Montreal*, 1988.

24 Frank P Ramsey. On a problem in formal logic. *Proceedings of the London Mathematical Society*, 30:264–286, 1930.

25 Andres J Ruiz-Vargas. Many disjoint edges in topological graphs. *Computational Geometry*, 62:1–13, 2017. doi:10.1016/J.COMGEO.2016.11.003.

26 Andrew Suk. Disjoint edges in complete topological graphs. *Discrete & Computational Geometry*, 49:280–286, 2013. doi:10.1007/S00454-012-9481-X.

27 Andrew Suk and Ji Zeng. Unavoidable Patterns in Complete Simple Topological Graphs. *Discrete & Computational Geometry*, 73:79–91, 2025. doi:10.1007/S00454-024-00658-6.

28 Gábor Tardos. Construction of locally plane graphs with many edges. In *Thirty Essays on Geometric Graph Theory*, pages 541–562. Springer, 2012.

29 Douglas R Woodall. Thrackles and deadlock. In *Combinatorial Mathematics and Its Applications*, pages 335–348, 1971.

30 Yian Xu. A new upper bound for Conway's thrackles. *Applied Mathematics and Computation*, 389:125573, 2021. doi:10.1016/J.AMC.2020.125573.

Gap-ETH-Tight Algorithms for Hyperbolic TSP and Steiner Tree

Sándor Kisfaludi-Bak
Department of Computer Science, Aalto University, Espoo, Finland

Saeed Odak
Department of Computer Science, Aalto University, Espoo, Finland

Satyam Singh
Department of Computer Science, Aalto University, Espoo, Finland

Geert van Wordragen
Department of Computer Science, Aalto University, Espoo, Finland

Abstract

We give an approximation scheme for the TSP in d-dimensional hyperbolic space that has optimal dependence on ε under Gap-ETH. For any fixed dimension $d \geq 2$ and for any $\varepsilon > 0$ our randomized algorithm gives a $(1+\varepsilon)$-approximation in time $2^{O(1/\varepsilon^{d-1})} n^{1+o(1)}$. We also provide an algorithm for the hyperbolic Steiner tree problem with the same running time.

Our algorithm is an Arora-style dynamic program based on a randomly shifted hierarchical decomposition. However, we introduce a new hierarchical decomposition called the hybrid hyperbolic quadtree to achieve the desired large-scale structure, which deviates significantly from the recently proposed hyperbolic quadtree of Kisfaludi-Bak and Van Wordragen (JoCG'25). Moreover, we have a new non-uniform portal placement, and our structure theorem employs a new weighted crossing analysis. We believe that these techniques could form the basis for further developments in geometric optimization in curved spaces.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Approximation algorithms analysis

Keywords and phrases Hyperbolic traveling salesman problem, TSP, Hyperbolic Steiner tree problem, Approximation scheme, Banyan, Hyperbolic geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.64

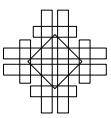
Related Version *Full Version*: `https://doi.org/10.48550/arXiv.2603.09834` [32]

Funding This work was supported by the Research Council of Finland, Grant 363444.

1 Introduction

The metric traveling salesman problem (TSP) is easily stated: given a set of n points in some metric space, find a cyclic permutation of points that minimizes the sum of lengths between consecutive points. A very commonly studied variant is when the underlying space is d-dimensional Euclidean space (henceforth denoted by $\mathbb{R}^d$), and admits $(1+\varepsilon)$-approximations in polynomial time (PTAS) for any fixed $\varepsilon > 0$ [1, 38]. The PTAS of Arora [1] for Euclidean TSP has been tirelessly improved, generalized, and optimized throughout the past decades by researchers working on geometric optimization. Today, we have Gap-ETH-tight running times in Euclidean space, and have working approximation schemes in any so-called doubling space to many related problems [3, 4, 5, 10, 31, 40, 42]. Moreover, many components of an Arora-style scheme can also be used in other important optimization problems related to network design, clustering, and facility location [2, 16, 17, 18, 20, 48, 22].


© Sándor Kisfaludi-Bak, Saeed Odak, Satyam Singh, and Geert van Wordragen;
licensed under Creative Commons License CC-BY 4.0



The key components of Arora's scheme [1] involve a hierarchical decomposition of the space (a quadtree) where the space is decomposed into fat[1] regions called *cells*, each of which in turn is decomposed into $f(d)$ smaller fat cells, where d is the dimension of the space. One can then argue that using small modifications, called *patching* on the tour, we can find a $(1+\varepsilon)$-approximate tour that crosses between neighboring cells of the decomposition only $g(\varepsilon, d)$ times (or, in some cases, $f(\varepsilon, d) \cdot \text{poly}(\log n)$ times), and these crossings come from a fixed set of points called *portals*. However, this argument cannot be made for a fixed hierarchical decomposition; one needs to apply a randomized decomposition and ensure that the number of crossings is small in expectation. Finally, one can employ a bottom-up dynamic programming algorithm on the hierarchy to compute partial solutions for each possible connection pattern of the cell's portals. Crucially, the running time depends on the number of portals used by the approximate tour as well as the number of children a cell can have, as both of these terms appear in the exponent of the dynamic programming algorithm.

In the Euclidean case, the hierarchical decomposition is a quadtree whose nodes correspond to axis-aligned hypercubes whose side length is a power of 2. The entire quadtree is shifted randomly, ensuring that in expectation the hypercube boundaries avoid hitting many short segments of the tour.

Doubling spaces are those metric spaces where for all $r \geq 0$ it holds that all balls of radius r can be covered by constantly many balls of radius $r/2$. The above techniques have been successfully generalized to this much more general setting, where randomized net-trees and padded decompositions take over the role of randomly shifted quadtrees [12, 15, 23, 28, 36, 44].

However, generalizing beyond the doubling space setting has been difficult. A natural target of extensions into the non-doubling setting is hyperbolic space, where the local structure is (almost) Euclidean, but at super-constant scale the space expands exponentially. Moreover, the exponential expansion leads to a tree-like behavior that can be exploited algorithmically.

Geometric optimization in hyperbolic space is still in its infancy, but there is reason to believe that even low-dimensional hyperbolic space is highly relevant from the practical perspective [7, 8, 11, 24, 25, 26, 39, 41, 43, 46, 47], and establishing the basic tools of geometric optimization in this setting will be widely useful for future algorithmic developments in both theory and practice. Thus, it would be prudent to explore the following problem.

▶ **Problem 1.** *Adapt algorithms from Euclidean geometric optimization to negatively curved (non-doubling) spaces, or prove lower bounds that rule out such adaptations.*

Krauthgamer and Lee [35] showed the existence of a PTAS for TSP in a more general setting: in so-called geodesic, visual and Gromov-hyperbolic spaces. This implies the existence of a PTAS in d-dimensional hyperbolic space (henceforth denoted by $\mathbb{H}^d$); however, the running time dependencies on d and ε are not explicit in their work. Their paper relies on doubling space results that bring factors $1/\varepsilon^{O(d)}$ to the exponent of the running time, thus Gap-ETH-tight running times cannot result from their approach. (Nonetheless, we include a more direct overview of their techniques and achieved running times to ours, see Section 1.2.)

In order to obtain tight running times, we need to make a more explicit analysis and not rely on generic doubling space results.

▶ **Problem 2.** *Is it possible to solve TSP in $\mathbb{H}^d$ as fast as in $\mathbb{R}^d$?*

A natural starting point is the recent work on hierarchical decompositions in hyperbolic space and the newly introduced hyperbolic quadtree of Kisfaludi-Bak and Van Wordragen [34].

[1] An object (a point set) is *α-fat* if the radius of the minimum circumscribed ball divided by the radius of the maximum inscribed ball is at least α. It is *fat* if it is α-fat for some fixed constant $\alpha > 0$.

In fact, García-Castellanos, Medbouhi, Marchetti, Bekkers, and Kragic [26] recently raised the question if a quadtree-based PTAS for hyperbolic Steiner tree is possible. We answer both the question of Problem 2 and of [26] affirmatively.

1.1 Our results

In this paper, we show that explicit Arora-style approximation schemes are possible in hyperbolic space, and we can match the efficiency of Euclidean approximation schemes apart from a factor of poly$(\log n)$.

▶ **Theorem 1.** *For any fixed $d \geq 2$ and any $\varepsilon > 0$, there is a randomized $(1+\varepsilon)$-approximation for TSP and Steiner tree in d-dimensional hyperbolic space in $2^{O(1/\varepsilon^{d-1})} n(\log n)^{2d(d-1)}$ time.*

To make our algorithm work, we need to introduce three new key components compared to earlier work in hyperbolic and Euclidean spaces:

- We need to change the hyperbolic quadtree of Kisfaludi-Bak and Van Wordragen [34] into a so-called hybrid hyperbolic quadtree (in short, hybrid tree). We create a hyperbolic shifting technique.
- We introduce a new portal placement for the cells of this hybrid tree. The portal placement is non-uniform, in stark contrast with all earlier algorithms.
- The proof of our structure theorem is built on a new amortized *weighted* crossing analysis.

Our techniques readily generalize to the Steiner tree problem and achieve the same running time. This resolves a recent question posed in [26] about hyperbolic Steiner tree. Our result can be considered as an important step toward geometric approximation schemes in non-doubling spaces of negative curvature, and hopefully a first step toward geometric optimization in more general (non-doubling) Riemannian manifolds.

It is important to note that due to our modified shifting technique, there is no polynomial time derandomization for this algorithm for general hyperbolic point sets, again deviating from the Euclidean setting. However, derandomization is possible in case the input has at most constant diameter, and it multiplies the running time by n^d in this case. Our derandomization can be stated as follows.

▶ **Corollary 2.** *For any fixed $d \geq 2$ and any $\varepsilon > 0$, there is a deterministic $(1+\varepsilon)$-approximation for TSP and Steiner tree in $\mathbb{H}^d$ in $2^{O(\operatorname{diam}(P)+1/\varepsilon^{d-1})} n^{d+1} (\log n)^{2d(d-1)}$ time, where $\operatorname{diam}(P)$ is the diameter of the input point set $P \subset \mathbb{H}^d$.*

The achieved dependence on ε is identical to the fastest Euclidean algorithm [31, 40], which is best possible under the so-called Gap Exponential Time Hypothesis (Gap-ETH) [21, 37]. Since a instance in $\mathbb{R}^d$ can be scaled to a small neighborhood and embedded in $\mathbb{H}^d$ with infinitesimally small distortion, any lower bound in $\mathbb{R}^d$ readily applies in $\mathbb{H}^d$.

▶ **Corollary 3** (Embedding the lower bound construction of [31])**.** *For any fixed integer $d \geq 2$ there is a $\gamma > 0$ such that there is no $2^{\gamma/\varepsilon^{d-1}} \operatorname{poly}(n)$ time $(1+\varepsilon)$-approximation algorithm for TSP in $\mathbb{H}^d$ under Gap-ETH.*

In particular, this means that our running time for TSP is Gap-ETH-tight for any constant d. However, it remains an open problem whether a *deterministic* running time of $2^{O(1/\varepsilon^{d-1})} n^{O(1)}$ is possible in $\mathbb{H}^d$ for point sets of super-constant diameter.

1.2 Overview of our techniques, comparison with earlier work

The techniques of Krauthgamer and Lee [35]. The main idea in the TSP algorithm of Krauthgamer and Lee [35] is to use a randomized embedding of the space into a distribution of trees. This exploits the tree-likeness of hyperbolic spaces at larger scales. At the same time, they propose an Arora-style approach inside balls of radius τ. Roughly, the space is partitioned into ball-like regions of radius τ in a randomized fashion, then embedded into a tree whose nodes correspond to partition classes.

The value of τ and the obtained running times are not explicit in [35]. Here we will briefly speculate about the resulting running time for the sake of comparison. When embedding into a tree with edges of length at least $\tau = \Omega_\delta(1/\varepsilon)$ one can observe that the tree distance and the true hyperbolic distance will differ by at most $O(\delta) = O(1)$, which is at most $(1 + O(\varepsilon))\tau$, i.e., measuring distances between different classes along the tree leads to only $1 + O(\varepsilon)$ multiplicative error.

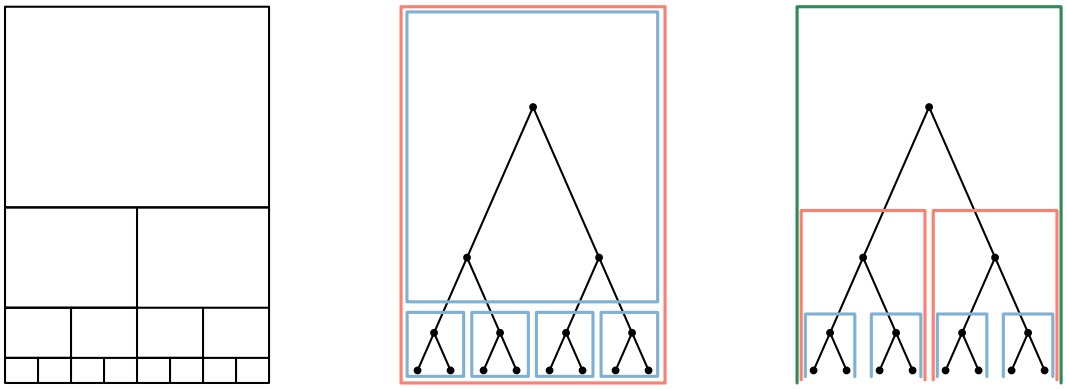
Within the partition classes of diameter τ, Krauthgamer and Lee must apply doubling space or Arora-style techniques [35]. However, even with newer doubling space algorithms, e.g. [5, 44], the doubling dimension of this neighborhood appears in the second level exponent. Since the doubling dimension of a τ-radius ball is *exponential in* τ, this leads to an at least triple-exponential algorithm in $1/\varepsilon$. Similarly, using some version of Arora's algorithm [1] on a distorted Euclidean instance would have to have the solution quality depend on the distortion of the τ-neighborhood, which is again exponential in τ. Thus, Arora's algorithm would have to be invoked with precision at least $\varepsilon' = 2^{-\tau}$, leading to a running time that is at least double-exponential in $\tau = \Omega(1/\varepsilon)$.

Overcoming exponential expansion. In a more explicit Arora-style algorithm, we have some challenges to overcome, in particular the following.

Ch1 Euclidean quadtrees (and net-trees in doubling spaces) have growth rates that depend only on the dimension, i.e., each cell has $f(d)$ children cells for some fixed function f. This is important as the dynamic program must take into account all portal placements of all children, and in particular, the running time has the number of children in the exponent. Due to the exponential expansion of hyperbolic spaces, no hierarchical decomposition matching the following properties is possible: (a) cells are fat, (b) cells of level ℓ have diameter $2^{\Theta(\ell)}$ and (c) each cell is partitioned into $f(d)$ children cells for some fixed function f. This issue was already observed by [34].

This property means that the hyperbolic quadtree of [34] is not suitable for our purposes: the number of children a cell can have is unbounded. Instead, we propose a new structure, called *hybrid hyperbolic quadtree*, or just *hybrid tree* for short, that has "open" cells at positive levels, and it fails the diameter property, but keeps fatness, and the number of children for cells is now bounded by 2^d. At the same time, we can still use the quadtree of [34] for its negative levels; these resemble a slightly distorted d-dimensional Euclidean quadtree, which is only defined now up to cells of diameter $O(1)$.

The binary tiling [13, 14] is a binary tree-like decomposition of the hyperbolic plane into isometric fat tiles. The tiling has a structure that resembles a binary tree: below each tile there are two other tiles, see the illustration in the half-plane model of $\mathbb{H}^2$ in Figure 1. For the sake of a simpler description, we compare the quadtree of [34] and our hybrid tree via this binary tree T. In the quadtree of [34], a cell of level ℓ will correspond to a subtree of T of height 2^ℓ, which can naturally be cut at half its depth into trees of height $2^{\ell-1}$, all the way until the tiling at level 0 is obtained consisting of subtrees of height 1 (i.e., the vertices of T). In our hybrid tree, a so-called *cell* of level $\ell > 0$ simply corresponds to a subtree of



(i) (ii) (iii)

Figure 1 (i) The binary tiling of $\mathbb{H}^2$ with isometric tiles. (ii) The cells of various positive levels of the corresponding quadtree illustrated on the binary tree T. (iii) Cells of the new hybrid tree of various positive levels illustrated on T.

height ℓ. Here we only allow subtrees whose leaves are also leaves of T. The corresponding cell diameter is not exponential in h, but each cell naturally has 2 children that are cells, plus a single child that is a cell of the binary tiling.

Notice that the resulting cells are "open" from the leaves' side, i.e., there is no cell below these cells. In particular, we place no portals on the bottom sides of cells of level $\ell \geq 1$.

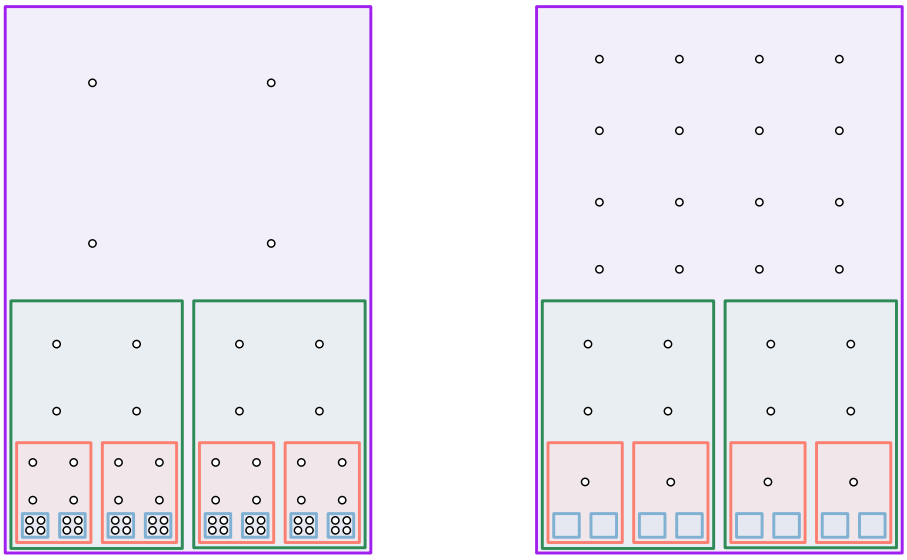
Our algorithm. Handling the hybrid tree in a black box fashion, our algorithm proceeds similarly to Arora's scheme. We first round our input points to centers of cells of size $\varepsilon \cdot \text{OPT}/n$. The level of these cells is $\ell_{\min}$, and it is useful to think of $\ell_{\min} < 0$ for this section.

We apply a random shift on the hybrid tree using the Euclidean vectors in the half-plane model. The shifting is very similar to the Euclidean shifting, with horizontal shifts corresponding to vertical cell boundaries in the bottom layer of the lowest level cells. On the vertical axis, the shifting has only constant range. While the shifting is easy to describe and somewhat natural, it is significantly different from the grid-based shifting in Euclidean space.

The main difficulty, as with all algorithms of this type, is the structure theorem: we need to prove that there exists a $(1+\varepsilon)$-approximate tour that intersects the cell boundaries only at a few pre-defined portals. In the Euclidean setting, one can prove that the number of intersections between the optimum tour and all grid hyperplanes is $O(\text{OPT})$ (assuming a minimum edge length of one), and then one can show that the patching cost incurred at each of these intersection points is $O(\varepsilon)$ in expectation. Since hyperbolic spaces are not vector spaces, most of the steps of the analysis fail and need new ideas: we already fail to bound the number of crossings in the same manner, as there is no convenient collection of hyperplanes that could play the role of grid hyperplanes in the Euclidean analysis.

The number of crossings, weighted portal analysis. In Arora's algorithm [1], grid lines parallel to each coordinate axis are analyzed in the same uniform manner. However, in a binary tiling, the vertical lines can intersect cells at larger heights and have a non-uniform behaviour. The main challenges with vertical lines (and in higher dimensions, vertical hyperplanes) are as follows:

Ch2 The vertical lines corresponding to our shifts can intersect the optimum tour an unbounded number of times, while in the Euclidean setting, the number of intersections with vertical grid lines is (up to constant factors) equal to the tour length.



(i) (ii)

Figure 2 Illustration of (i) uniform (naïve) portal placement and (ii) non-uniform portal placement, on a Side facet of a 3-dimensional cell of the hybrid tree. For simplicity, Figure (ii) uses a scaling factor of 2, although the true scaling factor is $2^{1-1/d}$. The empty circles represent the portals.

We overcome this via weighing the intersection points by the inverse of their z-coordinates. This allows us to have an absolute bound (for every shift) on the number of intersections between the tour and the vertical lines as a function of the length of the optimum tour.

Ch3 In dimension $d \geq 3$, a side facet of a cell is itself a cell whose structure resembles that of a binary tree. In particular, its surface area is an exponential function of its diameter. This is a very significant problem, as all known algorithms rely on placing portals in a grid-like fashion, ensuring that the distance from a grid point to the nearest portal is at most ε times the diameter of the facet. In Euclidean space, this requires the placement of $O(1/\varepsilon^{d-1})$ portals inside the hypercube-shaped facet. Here, this would lead to placing a number of portals that is exponential in $1/\varepsilon$, thus it would lead to a double-exponential running time as a function of $1/\varepsilon$. Since the number of portals is in the exponent of the running time of Arora-style algorithms, this would lead to a double-exponential dependence on $(1/\varepsilon)$.

We overcome this challenge via an elaborate portal placement. First, we observe that the probability that an intersection point ends up in a very large cell facet is exponentially small. Thus, we can afford to pay a large patching cost for certain crossings near the bottom of the bounding box with an exponentially small probability. Moreover, we must ensure that the patching cost gets progressively smaller as we get closer to the top of the side facets, as we need to counterbalance the weighted analysis used for the number of crossings. This leads to a non-uniform portal placement. For an illustration of the portal placement on a 2-dimensional Side facet of a 3-dimensional cell of the hybrid tree, see Figure 2(ii).

The dynamic programming: a banyan for hyperbolic Steiner tree. The dynamic programming algorithm for hyperbolic TSP can proceed in a bottom-up manner on the hybrid tree, as seen in previous algorithms [1, 31]. In fact, the same structure theorem readily works for

the Steiner tree problem, however, there is one difficulty: in the leaf cells (and in compressed cells) of the hybrid tree, the part of the optimum falling there is some Steiner forest, and such a forest can have Steiner points inside these cells. It is non-trivial to find these Steiner points in such a way as to preserve the approximation ratio. In the Euclidean setting, the standard solution is to use a so-called *banyan* of the portals of the cell.

A $(1+\varepsilon)$-*banyan* of the point set P is a geometric graph G on some set $P \cup Q$ such that for any subset $S \subseteq P$ there is a Steiner tree in G for S whose length is at most $1+\varepsilon$ times longer than the optimum Steiner tree of S in the ambient space. In the Euclidean setting of leaf cells, one can take Q to be a dense grid and G to be a complete geometric graph on $P \cup Q$: this will lead to a solution of size $\text{poly}(1/\varepsilon)$. However, due to the exponential volume growth in the hyperbolic setting, this is not a viable solution in hyperbolic space at larger distance scales: in this case, we show that it suffices to place potential Steiner points along all edges of a (d-dimensional) triangulation of portals and input points.

2 Preliminaries

Here we only give concise preliminaries for topics handled in the main text; for further details see the full version [32]. Let $\mathbb{H}^d$ and $\mathbb{R}^d$ denote d-dimensional hyperbolic space (of sectional curvature -1) and d-dimensional Euclidean space, respectively, with distance functions $\text{dist}_{\mathbb{H}}$ and $\text{dist}_{\mathbb{R}}$. For a given set of points $P = \{p_1, p_2, \ldots, p_n\}$, a tour is defined to be a cycle π that visits each point and returns to its starting point. Let $\text{len}_{\mathbb{R}}(\pi)$ denote its total Euclidean length and $\text{len}_{\mathbb{H}}(\pi)$ its total hyperbolic length. In the *hyperbolic Traveling Salesman problem*, the goal is to determine the hyperbolic length $\text{OPT}_{\mathbb{H}}$ of the shortest such tour passing through P. We also let $\text{OPT}_{\mathbb{R}}$ denote the Euclidean length of an optimal Euclidean tour. A *hyperbolic Steiner tree* of a point set $P \subseteq \mathbb{H}^d$ is a connected geometric graph whose vertex set includes P. The aim of the *hyperbolic Steiner Tree problem* is to compute a Steiner tree of P with minimum total hyperbolic length. Throughout this paper, we use log for the base-2 logarithm and ln for the natural logarithm. For a positive integer n, we use $[n]$ to denote the set $\{1, \ldots, n\}$. For $X \subset \mathbb{H}^d$, we write $\text{diam}(X)$ for the largest hyperbolic distance between any two points of X. For $\delta > 0$, a set N is a δ-*cover* of X if every $x \in X$ lies within distance δ of some $y \in N$.

Poincaré upper half-space model. The *Poincaré upper half-space model* maps hyperbolic space $\mathbb{H}^d$ to the Euclidean upper half-space, where each point $p \in \mathbb{H}^d$ is represented as $(x(p), z(p)) \in \mathbb{R}^{d-1} \times \mathbb{R}_{>0}$. In this model, the geodesic (shortest path) between p and q is either a vertical Euclidean line segment (when $x(p) = x(q)$), or an arc from a Euclidean circle that is perpendicular to the Euclidean hyperplane $z = 0$. The results in this paper are presented in the upper half-space model. We think of the z-direction as going "up" and the x-directions as going "sideways". For more details, see [14] or the textbooks [6, 29, 45].

Binary tiling. Let us first define the relevant notation from [34]. In the upper half-space model, an *axis-aligned horobox* B is a Euclidean axis-aligned box specified by its lexicographically minimal and maximal corner points $(x_{\min}(B), z_{\min}(B))$ and $(x_{\max}(B), z_{\max}(B))$. A *cube-based horobox* B is an axis-aligned horobox with $z_{\max}(B)/z_{\min}(B) = 2^h$ and $(x_{\max}(B) - x_{\min}(B))/z_{\min}(B) = (w, \ldots, w)$, where w is the *width* of B and h is the *height* of B.

To define the binary tiling, we begin with the cube-based horobox C such that $x_{\min}(C) = (0, \ldots, 0)$ and $z_{\min} = 1$, $w(C) = 1/\sqrt{d-1}$ and $h(C) = 1$. For $\sigma > 0$ and $\tau \in \mathbb{R}^{d-1}$, let $T_{\sigma,\tau}(x, z) = \sigma \cdot (x + \tau, z)$ be a family of transformations. The *binary tiling* is obtained by applying transformations $T_{\sigma,\tau}$, where $\sigma = 2^k$ for integers k, and $\tau \in \mathbb{Z}^{d-1}$, to the horobox C. This yields a tiling of $\mathbb{H}^d$ into isometric cells that are cube-based horoboxes; see Figure 1(i).

Euclidean tools from [1] and [31]. For small distances, our structure theorem relies on the Euclidean structure theorems of [1] and [31]. For the necessary definitions and the formal statement of these Euclidean structure theorems, we refer the reader to the full version of the paper [32] and to [1, 31].

3 Hybrid Hyperbolic Quadtree and Portals

The *hybrid hyperbolic quadtree* is a tree whose nodes, called cells, are cube-based horoboxes. Each cell is the disjoint union of its children in the tree and has an integer level such that same-level cells are (almost) isometric. A cell of level 0 is a cell of the binary tiling. A cell of level $\ell > 0$ is the union of its children, which are a cell C from the binary tiling and the 2^{d-1} cells of level $\ell - 1$ directly below C (for an illustration in $\mathbb{H}^2$, see Figure 1(iii)). Equivalently, a cell C of level $\ell \geq 0$ is a portion of the binary tiling resembling a complete 2^{d-1}-ary tree of height ℓ, whose subtrees of height $\ell' > 0$ are exactly the level-ℓ' descendants of C (see Figure 1(iii)). Note that all the cells of the same positive level are isometric. Cells of levels $\ell < 0$ are defined by splitting a binary tiling cell using Euclidean quadtree splits as in the hyperbolic quadtree of [34].

We can also define a *compressed hybrid hyperbolic quadtree* (shown in Figure 3) analogously to the compressed Euclidean quadtree; the details for this are in [32]. From this point onward, we refer to the hybrid hyperbolic quadtree as the *hybrid tree* and to the compressed hybrid hyperbolic quadtree as the *compressed hybrid tree*.

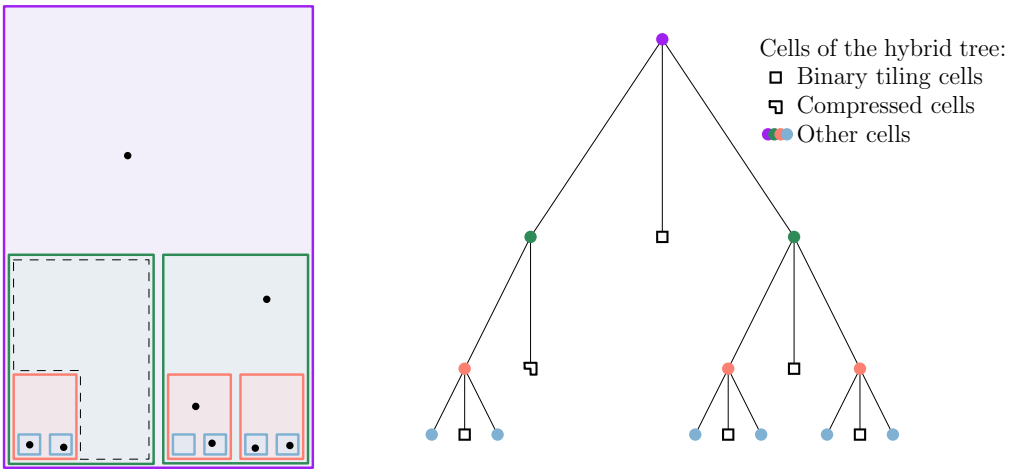


Figure 3 The compressed hybrid tree. (i) A point set and the cells of its compressed hybrid tree (A compressed cell is depicted in dashed lines). (ii) The tree structure of the hybrid tree.

Throughout the paper, the proofs of statements marked with ($\star$) can be found in the full version of the paper [32].

▶ **Lemma 4** ($\star$). *Let P be a set of n points in $\mathbb{H}^d$. There exists an $O(2^d n + dn \log n)$ time algorithm to build a compressed hybrid tree on P.*

Bounding box. Let B be the smallest Euclidean axis-aligned bounding box containing P. We first apply the same isometry $T_{\sigma,\tau}$ to both P and B so that $x_{\min}(B) = (0, \ldots, 0)$ and $z_{\min}(B) = 1$. Let s be the largest of the distances between non-adjacent facets of B.

▶ **Lemma 5** (⋆). $\operatorname{diam}(B) \leq (d+1)s$.

Since B is the smallest bounding box, $\textsc{opt}_{\mathbb{H}} \geq s$. Now, due to Lemma 5, $\textsc{opt}_{\mathbb{H}} \geq (1/(d+1)) \cdot \operatorname{diam}(B)$. Set $\ell_{\min}$ to be the minimum of 0 and the largest level where twice the diameter of a cell at that level is less than $(\varepsilon/(d+1)n) \cdot \operatorname{diam}(B)$. Define $\ell_{\max} - 1$ as the smallest hybrid tree level at which some cell C fully contains B, then let B^* denote the level $\ell_{\max}$ hybrid tree cell whose lexicographically minimal child cell is C.

Perturbation. Similar to Arora [1], our algorithm perturbs the instance by snapping each point in P to the center of level $\ell_{\min}$ cells (horobox) that contain them. Specifically, each point lying strictly inside a cell is mapped to the center of that cell, while points located on the boundaries of cells are assigned arbitrarily to the center of one of the adjacent cells. After snapping, we introduce an infinitesimal perturbation by adding the offset $\mu \cdot (1, 1, \ldots, 1)$ to each perturbed point in that cell, for an arbitrarily small $\mu > 0$. This ensures that, under every possible shift of the grid, no perturbed point lies exactly on a cell boundary.

▶ **Lemma 6** (⋆). *The length of an optimal tour over the perturbed points is at most* $(1+\varepsilon)\textsc{opt}_{\mathbb{H}}$.

3.1 Shifting of the hybrid tree

Similarly to Euclidean approximation schemes for TSP, we will use random shifting, though it is worth noting that our "shift" will not be an isometry. Let $\mathcal{C}_{\min}$ denote the cells of level $\ell_{\min}$ that lie in the lexicographically minimal child cell of B^*. We pick $\mathbf{a}_x \in \{x_{\min}(C) \mid C \in \mathcal{C}_{\min}\}$ and $\mathbf{a}_z \in \{z_{\min}(C) \mid C \in \mathcal{C}_{\min}\} \cap [1, 2)$ uniformly at random, then apply the transformation $(x, z) \mapsto (x - \mathbf{a}_x, z/\mathbf{a}_z)$ to the hybrid tree. This would be an isometry if $\mathbf{a}_z$ rescaled all coordinates; however, since it only rescales the z-coordinate, the distances are distorted by a factor of at most $\mathbf{a}_z \in [1, 2)$. Consequently, the transformed compressed hybrid tree has slightly different properties from the original hybrid tree, but we can still talk about cells and their children. We denote the shifted compressed hybrid tree under shift $\mathbf{a} = (\mathbf{a}_x, \mathbf{a}_z)$ by $Q_{\mathbf{a}}(P)$. We stop splitting once cells reach level $\ell_{\min}$, even if they still contain multiple points.

▶ **Lemma 7** (⋆). *Any level $\ell_{\min}$ cell of $Q_{\mathbf{a}}(P)$ contains at most 2^{d-1} distinct perturbed points.*

▶ **Lemma 8** (⋆). *Let C be a cell of level 0 in $Q_{\mathbf{a}}(P)$. There is a bijection $\phi : C \to [0, 1]^d$ such that $\phi(C)$ is a Euclidean unit hypercube that is split in $Q_{\mathbf{a}}(P)$ exactly as a Euclidean (compressed) quadtree. Additionally, ϕ is bi-Lipschitz: for any $p, q \in C$,*

$$\Omega(dist_{\mathbb{H}}(p, q)) \leq \operatorname{dist}_{\mathbb{R}}(\phi(p), \phi(q)) \leq O\Big(\sqrt{d} \cdot \operatorname{dist}_{\mathbb{H}}(p, q)\Big).$$

3.2 Portal placement

For any pair of sibling cells in $Q_{\mathbf{a}}(P)$, we now define a set of portals to be placed on their shared boundary. The placement depends on the level ℓ of these siblings.

Negative levels. When $\ell < 0$, we place the portals as in the Euclidean case [31] with a slight modification described below. In the Euclidean setting, the optimal tour consists of n line segments. Each segment crossing between two adjacent (Euclidean) quadtree cells must intersect their common facet. In the hyperbolic setting, however, tour segments can have endpoints in two adjacent negative-level cells but still avoid intersecting their common facet. To address this, we slightly modify the height of the common facet by extending it in the

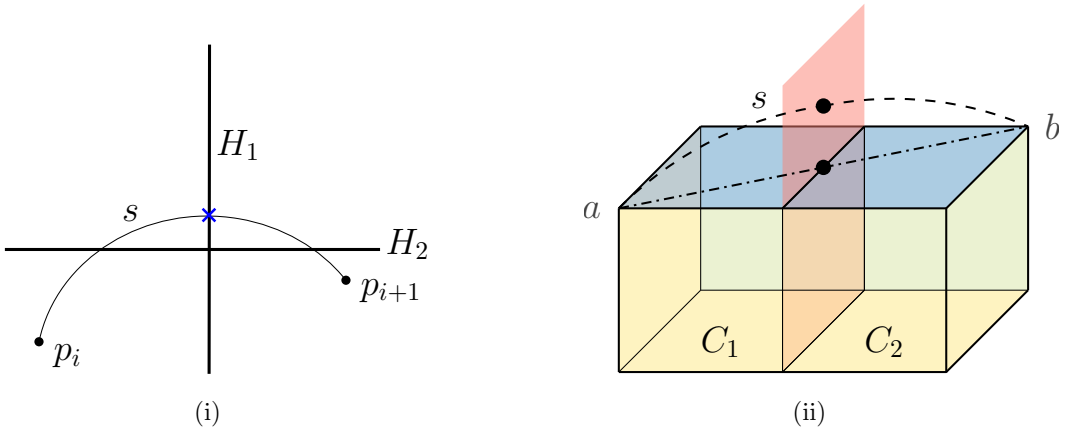


Figure 4 (i) The geodesic s has a crossing with the hyperplane H_1, but not with the hyperplane H_2: although s intersects H_2 twice, its endpoints lie on the same side. (ii) Two adjacent negative-level hybrid tree cells C_1 and C_2 share a common facet F. Extending F upward by a factor of 4 yields F', which is intersected by any geodesic s joining points $a \in C_1$ and $b \in C_2$.

z-direction. This ensures that every geodesic intersects the extended facet; see Figure 4(ii). Correspondingly, for a (hyperbolic or Euclidean) hyperplane H, we say here that a point $q \in H \cap \pi$ is a *crossing* of the tour π, if there exists $i \in [n]$ such that q lies on geodesic $p_i p_{i+1}$ and the endpoints p_i and p_{i+1} lie on different sides of H; see Figure 4(i).

▶ **Lemma 9** (⋆). *Let C_1 and C_2 be two adjacent level-ℓ ($\ell < 0$) hybrid tree cells sharing a common facet F of Euclidean height $\Delta = 2^{2^\ell} - 1$. If we extend this facet upward by a factor of 4, then every geodesic with endpoints in C_1 and C_2 intersects the extended facet F'.*

Non-negative levels. We now consider the case $\ell \geq 0$. Here, even the previously modified Euclidean portal placement cannot be applied directly, as it would result in an exponential blow-up in the number of portals and thus fail to achieve the desired result. To overcome this, we introduce a different portal placement strategy, described below. Note that a boundary facet shared between two sibling cells of level ℓ can be of two types:

Top: Facets that lie in horizontal Euclidean hyperplanes (hyperbolically these are horospheres). Each such facet is the top boundary of some binary tiling cell.

Side: Facets that lie in vertical hyperplanes. These are in fact $(d-1)$-dimensional cells, so each such facet corresponds to a cell in some $(d-1)$-dimensional hybrid tree.

Since Top facets come from binary tiling cells, Lemma 8 means we have constant distortion compared to Euclidean space. Consequently, we can simply construct a $(1/r)$-cover of the facet by placing a grid of $r^{d-1} \cdot d^{O(d)}$ equally-spaced portals.

For a Side facet σ, recall that σ is a cell of some $(d-1)$-dimensional hybrid tree and can therefore be seen as a complete 2^{d-2}-ary tree T_σ of binary tiling cells. We consider each cell F in T_σ individually and give a portal placement based on its (hop) distance h to the root of T_σ. Let $b = 2^{1-1/d}$. If $b^h \leq r$, we place portals according to a (b^h/r)-cover in F. Due to Lemma 8, this means we place $d^{O(d)} \cdot (r/b^h)^{d-1}$ portals. Otherwise, we do not place any portals. Now, the total number of portals placed for a Side facet is bounded by

$$\sum_{h=0}^{\infty} 2^{h(d-2)} \cdot d^{O(d)} \cdot (r/b^h)^{d-1} = d^{O(d)} \cdot r^{d-1} \sum_{h=0}^{\infty} 2^{-h/d} = d^{O(d)} \cdot r^{d-1}.$$

▶ **Observation 10.** *For any* Top *or* Side *facet, we place* $r^{d-1} \cdot d^{O(d)}$ *portals.*

4 Structure Theorem

Now we discuss and prove the structure theorem that allows us to develop a dynamic programming algorithm to solve the Hyperbolic TSP and Steiner Tree problem. We start by adapting the definitions for r-simple geometric graphs and r-simplifications from the Euclidean case [31] to our hyperbolic setting.

▶ **Definition 11** (r-simple geometric graph)**.** *Given a* $(d-1)$*-dimensional Euclidean box* F*, let* F^* *denote* F *with its* 2^{d-1} *corners removed. A geometric graph* π *is called* r-simple *if, for each facet* F *shared by a pair of sibling cells in* $Q_{\mathbf{a}}(P)$*,*

- *if* F *belongs to a non-negative level of the compressed hybrid tree, then the geometric graph* π *crosses* F *only through the portal points defined in* F*; or*
- *if* F *belongs to a negative level of the compressed hybrid tree, then either* π *crosses* F *at any number of its corner points and exactly one point of* F^**, or* π *crosses* F *only through the points from the uniform grid of size* g *on* F*, for some* $2^{d-1} \leq g \leq r^{2(d-1)}/|\pi \cap F^*|$*.*

▶ **Definition 12** (r-simplification)**.** *A geometric graph* π' *is called an* r-simplification *of a geometric graph* π *if* π' *is* r*-simple, and in each facet* F *of a negative level where* $|\pi \cap F^*| = 1$*, the single non-corner crossing of* π' *is the point in* $\pi \cap F^*$*.*

Having these definitions, we are now ready to state our main theorem.

▶ **Theorem 13** (hyperbolic structure theorem)**.** *Let* $\boldsymbol{a}$ *be a random shift and let* π *be a tour (or Steiner tree) of the point set* $P \subset \mathbb{H}^d$*. Then for* $r > 0$*, there is a tour (resp. Steiner tree)* π' *of* P *that is an* r*-simplification of* π *such that*

$$\mathbb{E}_{\mathbf{a}}[\mathrm{len}_{\mathbb{H}}(\pi') - \mathrm{len}_{\mathbb{H}}(\pi)] \leq O(d^3) \cdot \frac{\mathrm{len}_{\mathbb{H}}(\pi)}{r}.$$

In order to prove Theorem 13, we start by proving a bound on the patching cost of the negative levels of the compressed hybrid tree. Note that we state and prove everything for a tour, but every step works identically for a Steiner tree.

▶ **Lemma 14** ($\star$)**.** *Let* π_C *be the part of a tour* π *inside a cell* C *of the binary tiling. Then there exists a tour* π'_C *in* C *containing* $C \cap P$ *and connecting* $a, b \in \partial C$ *with a path if and only if* $a, b \in \partial C$ *are connected in* π_C *such that* π'_C *is either*

- *an* r*-simplification with respect to the negative levels and patched to the portals of [31] or*
- r*-simple with respect to the negative levels and patched to the portals of Arora [1],*

and π'_C *has expected length* $(1 + O(d^3/r))\mathrm{len}_{\mathbb{H}}(\pi_C)$*.*

Now, we prove our structure theorem.

Proof of Theorem 13. We describe a patching procedure for negative and non-negative levels of our shifted compressed hybrid tree separately.

Patching negative levels. We first modify the optimum tour π to get a tour π_1 that is slightly longer in expectation, and only crosses the shared boundaries of negative-level sibling cells through portals. In each binary tiling cell C, we apply Lemma 14 on $\pi_C = \pi \cap C$. As a result, we get a collection π'_C of paths in C whose expected length is $(1+O(d^3/r))\mathrm{len}_{\mathbb{H}}(\pi \cap C)$. Let π_1 denote the union of π'_C over all binary tiling cells C. Notice that $\pi_1 \cap C$ connects two points of ∂C if and only if the same points are connected by π, and π_1 covers the same set of input points; thus π_1 is a tour of P of expected length $(1 + O(d^3/r))\mathrm{len}_{\mathbb{H}}(\pi)$.

Patching non-negative levels. We further modify the tour π_1 to get a tour π_2 that is slightly longer in expectation, and only crosses the cell boundaries between non-negative level siblings of the compressed hybrid tree through portals. This can be achieved as follows. For each crossing point x of π_1 with a shared facet F of cell siblings of level at least 0, we add two hyperbolic segments on each side of the facet connecting a copy of x and the portal that is closest to x. (Technically, the segments are added at some infinitesimal distance from the facet.) This operation transfers all crossings to the nearest portals. Next, if some portal v is used at least three times by the tour, then by the patching lemma [1, Lemma 3] the tour can be simplified to go through the portal at most twice.

Patching costs on non-negative levels. To bound the patching cost, we want to give deterministic bounds on the number of intersections between the tour π and the hyperplanes used by the cells at level $\ell_{\min}$, then combine this with the expected cost for each intersection under random shifts $\mathbf{a}$. We do this separately for the vertical hyperplanes and the horizontal ones, since their behavior and portal placement differ.

Let $\mathcal{V}$ be the set of all vertical hyperplanes obtained by extending the vertical facets of the cells at level $\ell_{\min}$ in the unshifted hybrid tree. Let δ be the (maximum) diameter of all level $\ell_{\min}$ cells. Since these cells are $\Omega(1/\sqrt{d})$-fat [34, Theorem 7], we observe the following.

▶ **Observation 15.** *After perturbation, the distance between two distinct points is* $\Omega(\delta/\sqrt{d})$.

Decompose π into consecutive segments $\pi^1, \dots, \pi^n$. Let I denote the set of intersection points of π and all hyperplanes in $\mathcal{V}$. To bound the patching cost of the vertical intersections of the tour π, we use a weighted portal analysis. We assign each $q \in I$ a weight equal to $1/z(q)$. To bound the expected patching cost of the Side facets, we start by bounding the total weight of intersections in I as follows.

▷ Claim 16. $\sum_{p\in I} \frac{1}{z(p)} \le O\Big(\frac{d\sqrt{d}}{\delta} \cdot \mathrm{len}_{\mathbb{H}}(\pi)\Big)$.

Proof. We consider each of the $d-1$ horizontal axes separately; let $\mathcal{V}'$ be a set of hyperplanes perpendicular to one of these axes. For each $j \in [n]$, let $I^j = \{p_1^j, \dots, p_{k_j}^j\}$ be the ordered set of intersection points of π^j with $\mathcal{V}'$, and $I' = \bigcup_{j\in[n]} I^j$. For $i \in [k_j - 1]$, let $z_i^j = \min\{z(p_i^j), z(p_{i+1}^j)\}$. The Euclidean distance between adjacent hyperplanes from $\mathcal{V}'$ is given by the horobox width of level-$\ell_{\min}$ cells, which is $w \coloneqq \Theta(\delta/\sqrt{d})$ according to [34, Lemma 6]. By [34, Observation 11], the distance from a point $(w, 0, \dots, 0, z_i^j)$ to hyperplane $x_1 = 0$ is $\operatorname{arsinh}\frac{w}{z_i^j}$. Since $\operatorname{arsinh} x = \Omega(x)$ for $x = O(1)$, $\mathrm{len}_{\mathbb{H}}(p_i^j p_{i+1}^j) = \Omega\Big(\frac{\delta}{z_i^j\sqrt{d}}\Big)$. Summing this over $i \in [k_j - 1]$ yields

$$\mathrm{len}_{\mathbb{H}}(\pi^j) \ge \sum_{i\in[k_j-1]} \mathrm{len}_{\mathbb{H}}(p_i^j p_{i+1}^j) \ge \sum_{i\in[k_j-1]} \Omega\left(\frac{\delta}{z_i^j\sqrt{d}}\right).$$

Since $1/(z_i^j) = \max\{1/z(p_i^j), 1/z(p_{i+1}^j)\}$, we have $\sum_{i\in[k_j-1]} \frac{1}{z_i^j} \ge \frac{1}{2}\sum_{p\in I^j} \frac{1}{z(p)}$. Hence, $\sum_{p\in I^j} \Omega\Big(\frac{\delta}{\sqrt{d}\cdot z(p)}\Big)$ is a lower bound for $\mathrm{len}_{\mathbb{H}}(\pi^j)$. If $k_j \le 1$ (that is π^j intersects $\mathcal{V}'$ at most once), the lower bound follows from Observation 15. Summing over $j \in [n]$, we have

$$\mathrm{len}_{\mathbb{H}}(\pi) \ge \sum_{j\in[n]} \sum_{p\in I^j} \Omega\left(\frac{\delta}{\sqrt{d}\cdot z(p)}\right) = \frac{\delta}{\sqrt{d}} \sum_{p\in I'} \Omega\left(\frac{1}{z(p)}\right).$$

Hence, $\sum_{p\in I'} \frac{1}{z(p)} \le O\Big(\frac{\sqrt{d}}{\delta} \cdot \mathrm{len}_{\mathbb{H}}(\pi)\Big)$ and summing over all axes gives the claim. ◁

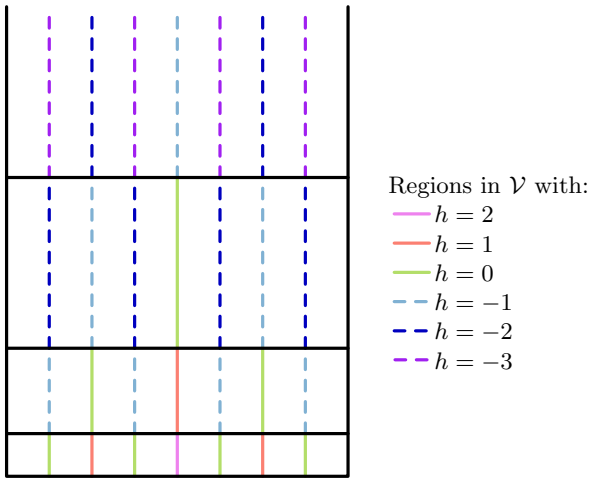


Figure 5 Distribution of the value h used in Claim 17 (here $\ell_{\min} = 0$). Note that the solid lines give the binary tiling.

Next, we state a bound for the expected patching cost of vertical intersections.

▷ Claim 17. Let $p \in I$. Then $\mathbb{E}_{\mathbf{a}}[\text{patching cost for } p] = O\left(\frac{d \cdot \delta}{r \cdot z(p)}\right)$.

Proof. For any vertical hyperplane $V \in \mathcal{V}$, partition it into regions $(V_h)_{h \in \mathbb{Z}}$ with the horizontal Euclidean hyperplanes $z = 2^i/\mathbf{a}_z$ (for all $i \in \mathbb{Z}$, unrelated to h) used by the shifted binary tiling. Here, V_0 denotes the highest region of V that still separates two cells from the shifted binary tiling, and V_{h+1} is the region directly below region V_h, as shown in Figure 5. Note that for $h \geq 0$, any two regions V_h and V'_h (with $V \neq V'$ but the same h) have the same portal placement: if $b^h \leq r$ the portals form a (b^h/r)-cover and otherwise there are no portals. For the given point p, we now let the random variable H be such that $p \in V_H$ for some $V \in \mathcal{V}$, and calculate the expected patching cost based on the behavior of H.

Note that H depends on the shift $\mathbf{a}$ as well as $z(p)$. Let us first determine the probability distribution of H at the bottom row $z(p) = 1/\mathbf{a}_z$. Note that the number of shifts $\mathbf{a}_x$ giving $H = h$ is twice the number giving $H = h + 1$. Thus, if we have k shifts that give $H = \ell_{\min}$ then $\lfloor k \cdot 2^{\ell_{\min} - h - 1} \rfloor$ shifts give $H = h$. Seeing as we have $2k - 1$ shifts in total,

$$\mathbb{P}(H = h) = O(2^{\ell_{\min} - h}) = O(\delta/2^h).$$

For general $z(p)$, note that H decreases by one every time $z(p)$ doubles and hence

$$\mathbb{P}(H = h) = O\left(\delta/2^{h + \log \frac{z(p)}{\mathbf{a}_z}}\right) = O\left(\frac{\delta}{2^h z(p)}\right),$$

since $\mathbf{a}_z \in [1, 2)$. For convenience, let $t = O(\delta/z(p))$ to simplify this to $\mathbb{P}(H = h) \leq t/2^h$.

When $H \geq 0$ and $b^H \leq r$, the patching cost will be at most $2b^H/r$. If $b^H > r$, we can still route through some other portal on the facet, which will be within distance $O(H - \log_b r)$. If $H < 0$, then the patching cost is zero. Therefore, we have

$$\begin{aligned}
\mathbb{E}_{\mathbf{a}}[\text{patching cost for } p] &\leq \sum_{h=0}^{\log_b r} 2b^h t/(r\cdot 2^h) + \sum_{h=\log_b r}^{\infty} O(h-\log_b r)\cdot t/2^h \\
&< \frac{2t}{r}\sum_{h=0}^{\infty}(b/2)^h + O(t/2^{\log_b r})\sum_{h=0}^{\infty} h/2^h \\
&= \frac{2t}{r}\cdot\frac{1}{1-b/2} + O(t/2^{\log_b r}) \\
&= \frac{2t}{r}\cdot\frac{1}{1-2^{-1/d}} + O(t/r^{\frac{1}{1-1/d}}) \\
&= O(dt/r). \\
&\qquad\qquad \text{(since } (1+x)^r \leq 1+rx \text{ for reals } 0\leq r\leq 1 \text{ and } x\geq -1.)
\end{aligned}$$

Hence, the claim follows. ◁

Consider the crossings of π_1 with Side facets. By Claim 16 and Claim 17, we have

$$\sum_{p\in I}\mathbb{E}_{\mathbf{a}}[\text{patching cost for } p] = O\left(\frac{d\delta}{r}\right)\sum_{p\in I}\frac{1}{z(p)} = O\left(\frac{d^2\sqrt{d}}{r}\right)\text{len}_{\mathbb{H}}(\pi_1).$$

To bound the number of crossings with Top facets, we have the following claim.

▷ Claim 18 (⋆). Let Λ be the number of intersections of π with Top facets of non-negative level cells. We have $\mathbb{E}_{\mathbf{a}}[\Lambda] = O(\text{len}_{\mathbb{H}}(\pi)\sqrt{d})$.

For Top facets, the patching cost is always at most $2/r$ by construction, thus by Claim 18 their expected total patching cost is $O\left(\frac{\sqrt{d}}{r}\right)\text{len}_{\mathbb{H}}(\pi_1)$. Combining the two contributions, the expected total patching cost is $O\left(\frac{d^2\sqrt{d}}{r}\right)\text{len}_{\mathbb{H}}(\pi_1)$. Finally, using $\text{len}_{\mathbb{H}}(\pi_1) = (1+O(1/r))\text{len}_{\mathbb{H}}(\pi)$ to express the bound in terms of π, we obtain $\mathbb{E}_{\mathbf{a}}\left[\text{len}_{\mathbb{H}}(\pi_2) - \text{len}_{\mathbb{H}}(\pi_1)\right] = O\left(\frac{d^2\sqrt{d}}{r}\right)\text{len}_{\mathbb{H}}(\pi)$, as required. We conclude the proof by observing that $\pi' := \pi_2$ is an r-simplification of π and has the desired length. ◀

5 Conclusion

In this paper we have obtained approximation schemes for d-dimensional hyperbolic TSP and hyperbolic Steiner tree with running time $2^{O(1/\varepsilon^{d-1})}n\,\text{poly}(\log n)$ (our approximation schemes follow from the structure theorem, mostly using standard techniques; we go into this in the full version [32]). Some interesting open questions remain for future work.

- The reason why we do not yet match the running times of [31] is because the faster algorithms rely on light Euclidean spanners, while no light (Steiner) spanners exist for hyperbolic point sets as of yet. Are there Steiner spanners of lightness $\text{poly}(1/\varepsilon)$ in $\mathbb{H}^d$? What about light banyans?
- Is it possible to use the techniques of [40] to get a linear dependence on n?
- Is there a deterministic algorithm for hyperbolic TSP without dependence on the diameter of the input?
- It has been observed on several occasions that the complexity of optimization problems in hyperbolic space decreases as points are at larger distance from each other [9, 30, 33]. In line with this, one can embed any hyperbolic point set into an edge-weighted tree

with additive error $O(\log n)$ [27, Chapter 6] in time $O(n^2)$ [19], which already implies an $O(n^2)$ time TSP algorithm when $\text{OPT}_{\mathbb{H}} = \Omega(n \log n/\varepsilon)$. Can we already achieve milder dependence on ε for smaller $\text{OPT}_{\mathbb{H}}$, or keep near-linear dependence on n?

References

1 Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

2 Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proceedings of the 13th ACM Symposium on the Theory of Computing, (STOC)*, pages 106–113. ACM, 1998. doi:10.1145/276698.276718.

3 Yair Bartal and Lee-Ad Gottlieb. A linear time approximation scheme for Euclidean TSP. In *Proceedings of the 54th IEEE Symposium on Foundations of Computer Science, (FOCS)*, pages 698–706. IEEE, 2013. doi:10.1109/FOCS.2013.80.

4 Yair Bartal and Lee-Ad Gottlieb. Near-linear time approximation schemes for Steiner tree and forest in low-dimensional spaces. In *Proceedings of the 53rd ACM SIGACT Symposium on Theory of Computing, (STOC)*, pages 1028–1041. ACM, 2021. doi:10.1145/3406325.3451063.

5 Yair Bartal, Lee-Ad Gottlieb, and Robert Krauthgamer. The traveling salesman problem: Low-dimensionality implies a polynomial time approximation scheme. *SIAM J. Comput.*, 45(4):1563–1581, 2016. doi:10.1137/130913328.

6 Riccardo Benedetti and Carlo Petronio. *Lectures on Hyperbolic Geometry*. Springer Science & Business Media, 1992. doi:10.1007/978-3-642-58158-8.

7 Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, Ulrich Meyer, Manuel Penschuck, and Christopher Weyand. Efficiently generating geometric inhomogeneous and hyperbolic random graphs. *Network Science*, 10(4):361–380, 2022. doi:10.1017/nws.2022.32.

8 Thomas Bläsius, Tobias Friedrich, and Anton Krohmer. Hyperbolic random graphs: Separators and treewidth. In *Proceedings of the 24th European Symposium on Algorithms, (ESA)*, volume 57 of *LIPIcs*, pages 15:1–15:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.15.

9 Thomas Bläsius, Jean-Pierre von der Heydt, Sándor Kisfaludi-Bak, Marcus Wilhelm, and Geert van Wordragen. Structure and independence in hyperbolic uniform disk graphs. In *Proceedings of the 41st International Symposium on Computational Geometry, (SoCG)*, volume 332 of *LIPIcs*, pages 21:1–21:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SOCG.2025.21.

10 Glencora Borradaile, Philip N. Klein, and Claire Mathieu. A polynomial-time approximation scheme for Euclidean Steiner forest. *ACM Trans. Algorithms*, 11(3):19:1–19:20, 2015. doi:10.1145/2629654.

11 Karl Bringmann, Ralph Keusch, and Johannes Lengler. Geometric inhomogeneous random graphs. *Theor. Comput. Sci.*, 760:35–54, 2019. doi:10.1016/j.tcs.2018.08.014.

12 Costas Busch, Da Qi Chen, Arnold Filtser, Daniel Hathcock, D. Ellis Hershkowitz, and Rajmohan Rajaraman. One tree to rule them all: Poly-logarithmic universal steiner tree. In *Proceedings of the 64th IEEE Symposium on Foundations of Computer Science, (FOCS)*, pages 60–76. IEEE, 2023. doi:10.1109/FOCS57990.2023.00012.

13 Károly Böröczky. Gömbkitöltések állandó görbületű terekben I. *Matematikai Lapok (in Hungarian)*, 25(3-4):265–306, 1974.

14 James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. Hyperbolic geometry. *Flavors of geometry*, 31(59-115):2, 1997.

15 T.-H. Hubert Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics. *ACM Trans. Algorithms*, 12(4):55:1–55:22, 2016. doi:10.1145/2915183.

16 Vincent Cohen-Addad. Approximation schemes for capacitated clustering in doubling metrics. In *Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2241–2259. SIAM, 2020. doi:10.1137/1.9781611975994.138.

17 Vincent Cohen-Addad, Andreas Emil Feldmann, and David Saulpic. Near-linear time approximation schemes for clustering in doubling metrics. *J. ACM*, 68(6):44:1–44:34, 2021. doi:10.1145/3477541.

18 Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. *SIAM Journal on Computing*, 48(2):644–667, 2019. Preliminary version in FOCS 2016. doi:10.1137/17M112717X.

19 Anders Cornet and Eduardo Martínez-Pedroza. Gromov's approximating tree and the all-pairs bottleneck paths problem. *CoRR*, 2025. doi:10.48550/arXiv.2408.05338.

20 Artur Czumaj, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Pavel Veselý. Streaming algorithms for geometric steiner forest. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming, (ICALP)*, volume 229 of *LIPIcs*, pages 47:1–47:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.47.

21 Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *Electron. Colloquium Comput. Complex.*, 23:128, 2016. URL: http://eccc.hpi-web.de/report/2016/128.

22 Jittat Fakcharoenphol and Nonthaphat Wongwattanakij. A PTAS for k-hop MST on the euclidean plane: Improving dependency on k. *Information Processing Letters*, 190:106581, 2025. doi:10.1016/j.ipl.2025.106581.

23 Arnold Filtser. On strong diameter padded decompositions. In *Proceedings of the Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, (APPROX/RANDOM)*, volume 145 of *LIPIcs*, pages 6:1–6:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.6.

24 Tobias Friedrich and Anton Krohmer. On the diameter of hyperbolic random graphs. *SIAM J. Discret. Math.*, 32(2):1314–1334, 2018. doi:10.1137/17M1123961.

25 Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Proceedings of the Advances in Neural Information Processing Systems, (NeurIPS)*, pages 5350–5360, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/dbab2adc8f9d078009ee3fa810bea142-Abstract.html.

26 Alejandro García-Castellanos, Aniss Aiman Medbouhi, Giovanni Luca Marchetti, Erik J. Bekkers, and Danica Kragic. HyperSteiner: Computing heuristic hyperbolic Steiner minimal trees. In *Proceedings of the 2025 Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 194–208. SIAM, 2025. doi:10.1137/1.9781611978339.16.

27 M. Gromov. *Hyperbolic Groups*, pages 75–263. Springer, 1987. doi:10.1007/978-1-4613-9586-7_3.

28 Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the 44th Symposium on Foundations of Computer Science, (FOCS)*, pages 534–543. IEEE, 2003. doi:10.1109/SFCS.2003.1238226.

29 Birger Iversen. *Hyperbolic Geometry*, volume 25 of *London Mathematical Society Student Texts*. Cambridge University Press, 1992. doi:10.1017/CBO9780511569333.

30 Sándor Kisfaludi-Bak. A quasi-polynomial algorithm for well-spaced hyperbolic TSP. *J. Comput. Geom.*, 12(2):25–54, 2021. doi:10.20382/JOCG.V12I2A3.

31 Sándor Kisfaludi-Bak, Jesper Nederlof, and Karol Węgrzycki. A gap-ETH-tight approximation scheme for Euclidean TSP. *J. ACM*, 2025. doi:10.1145/3766548.

32 Sándor Kisfaludi-Bak, Saeed Odak, Satyam Singh, and Geert van Wordragen. Gap-ETH-tight algorithms for hyperbolic TSP and Steiner tree. *CoRR*, 2026. doi:10.48550/arXiv.2603.09834.

33 Sándor Kisfaludi-Bak and Geert van Wordragen. Near-optimal dynamic Steiner spanners for constant-curvature spaces. *CoRR*, 2025. doi:10.48550/arXiv.2509.01443.

34 Sándor Kisfaludi-Bak and Geert van Wordragen. A quadtree, a Steiner spanner, and approximate nearest neighbours in hyperbolic space. *Journal of Computational Geometry*, 16(2):145–174, 2025. doi:10.20382/jocg.v16i2a5.

35 Robert Krauthgamer and James R. Lee. Algorithms on negatively curved spaces. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science, (FOCS)*, pages 119–132. IEEE, 2006. doi:10.1109/FOCS.2006.9.

36 An La and Hung Le. Dynamic locality sensitive orderings in doubling metrics. In *Proceedings of the 57th ACM Symposium on Theory of Computing, (STOC)*, pages 1178–1189. ACM, 2025. doi:10.1145/3717823.3718209.

37 Pasin Manurangsi and Prasad Raghavendra. A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming, (ICALP)*, volume 80 of *LIPIcs*, pages 78:1–78:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.78.

38 Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.

39 Karim Mnasri, Bhilahari Jeevanesan, and Jörg Schmalian. Critical phenomena in hyperbolic space. *Physical Review B*, 92:134423, 2015. URL: https://api.semanticscholar.org/CorpusID:118004293.

40 Tobias Mömke and Hang Zhou. A linear time gap-ETH-tight approximation scheme for Euclidean TSP. *CoRR*, 2025. arXiv:2411.02585.

41 Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 3776–3785. PMLR, 2018. URL: http://proceedings.mlr.press/v80/nickel18a.html.

42 Satish Rao and Warren D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proceedings of the 13th ACM Symposium on the Theory of Computing, (STOC)*, pages 540–550. ACM, 1998. doi:10.1145/276698.276868.

43 Yuval Shavitt and Tomer Tankel. On the curvature of the internet and its usage for overlay construction and distance estimation. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*. IEEE, 2004. doi:10.1109/INFCOM.2004.1354510.

44 Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the 36th ACM Symposium on Theory of Computing, (STOC)*, pages 281–290. ACM, 2004. doi:10.1145/1007352.1007399.

45 William P. Thurston. *Three-Dimensional Geometry and Topology, Volume 1*. Princeton University Press, 1997. doi:10.1515/9781400865321.

46 Abraham A. Ungar. Einstein's special relativity: The hyperbolic geometric viewpoint. *CoRR*, 2013. arXiv:1302.6961.

47 Kevin Verbeek and Subhash Suri. Metric embedding, hyperbolic space, and social networks. In *Proceedings of the 30th Symposium on Computational Geometry, SoCG*, page 501. ACM, 2014. doi:10.1145/2582112.2582139.

48 Di Wu, Qilong Feng, and Jianxin Wang. Approximation algorithms for fair k-median problem without fairness violation. *Theor. Comput. Sci.*, 985:114332, 2024. doi:10.1016/j.tcs.2023.114332.

Near-Optimal Dynamic Steiner Spanners for Constant-Curvature Spaces

Sándor Kisfaludi-Bak
Aalto University, Espoo, Finland

Geert van Wordragen
Aalto University, Espoo, Finland

Abstract

We consider Steiner spanners in Euclidean and non-Euclidean geometries. In the Euclidean setting, a recent line of work initiated by Le and Solomon [FOCS'19] and further improved by Chang et al. [SoCG'24] obtained Steiner $(1+\varepsilon)$-spanners of size $O_d(\varepsilon^{(1-d)/2}\log(1/\varepsilon)n)$, nearly matching the lower bound $\Omega_d(\varepsilon^{(1-d)/2}n)$ of Bhore and Tóth [SIDMA'22].

We obtain Steiner $(1+\varepsilon)$-spanners of size $O_d(\varepsilon^{(1-d)/2}\log(1/\varepsilon)n)$ not only in d-dimensional Euclidean space, but also in d-dimensional spherical and hyperbolic space. For any fixed dimension d, the obtained edge count is optimal up to an $O(\log(1/\varepsilon))$ factor in each of these spaces. Unlike earlier constructions, our Steiner spanners are based on simple quadtrees, and they can be dynamically maintained, leading to efficient data structures for dynamic approximate nearest neighbours and bichromatic closest pair.

In the hyperbolic setting, we also show that 2-spanners in the hyperbolic plane must have $\Omega(n\log n)$ edges, and we obtain a 2-spanner of size $O_d(n\log n)$ in d-dimensional hyperbolic space, matching our lower bound for any constant d. Finally, we give a Steiner spanner with *additive* error ε in hyperbolic space with $O_d(\varepsilon^{(1-d)/2}\log(\alpha(n)/\varepsilon)n)$ edges, where $\alpha(n)$ is the inverse Ackermann function.

Our techniques generalize to closed orientable surfaces of constant curvature as well as to some other quotient spaces.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Approximation algorithms analysis; Theory of computation → Nearest neighbor algorithms

Keywords and phrases hyperbolic geometry, Steiner spanner, dynamic approximate nearest neighbours

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.65

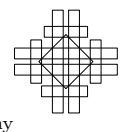
Related Version *Full Version*: `https://arxiv.org/abs/2509.01443` [40]

Funding *Sándor Kisfaludi-Bak*: Supported by the Research Council of Finland, Grant 363444.

1 Introduction

Given a set of points in some metric space, what is the most efficient way to represent all pairwise distances? One can use a complete geometric graph: the vertices are the given points, and the edges are weighted by the corresponding distance. A *t-spanner* is some small subgraph G of this complete graph such that the shortest path in G among any pair of points is at most t times longer than their distance. In Euclidean space, one can obtain $(1+\varepsilon)$-spanners of linear size for any fixed constant $\varepsilon > 0$ [20, 35]. Spanners are not only interesting in their own right, but they are also used in a plethora of algorithmic applications, and they lie at the heart of many approximation algorithms [54, 41, 24, 3, 39]. There is an ongoing research effort to study spanners in more general settings, and to establish optimal trade-offs between their stretch ($t = 1+\varepsilon$) and their size (edge count).


© Sándor Kisfaludi-Bak and Geert van Wordragen;
licensed under Creative Commons License CC-BY 4.0




Spanners and dynamic spanners have been studied extensively in Euclidean spaces [49, 19, 2, 21] as well as in the more general setting of spaces with bounded doubling dimension [28, 30, 13, 56, 14]. The *doubling dimension* of a metric space is the minimum k such that any ball in the metric space can be covered by 2^k balls of half its radius. Today, we understand the optimal trade-offs between stretch and spanner size in Euclidean and doubling spaces. In Euclidean space, several techniques for spanner constructions, such as greedy spanners [1, 49] and locality-sensitive orderings [15, 29] yield $(1+\varepsilon)$-spanners of size (near) $\mathcal{O}_d(\varepsilon^{1-d}n)$, which is optimal for any fixed dimension d [47]. Well-separated pair decompositions (WSPDs) [10] give a slightly worse edge count of $\mathcal{O}_d(\varepsilon^{-d}n)$, but have a simpler construction and analysis.

Many of these techniques have been ported to doubling spaces, and have been used successfully to create spanners (and approximation algorithms). Net trees [33] play the role of a hierarchical decomposition of the space rather than quadtrees, and one can build WSPDs [59, 33], greedy spanners [58], and locality-sensitive orderings [15, 26, 44]. Gottlieb and Roditty [31] provide dynamic spanners in doubling spaces of dimension d with $\varepsilon^{-\mathcal{O}(d)}n$ edges and update time $\varepsilon^{-\mathcal{O}(d)}\log n$. Note here that the doubling dimension of Euclidean space is already (a constant fraction) more than d which makes it difficult to match Euclidean results, and moreover spanners have a worse lower bound of $\Omega_d(\varepsilon^{-d}n)$ edges [45].

While the above spanners are already useful, one can obtain even smaller (sparser) spanners if we allow so-called Steiner points in the spanner graph, that is, new points from the ambient space. More precisely, a *Steiner* $(1+\varepsilon)$*-spanner* of a point set P in a metric space (X, dist) is a subgraph G of the complete geometric graph on $P \cup S$ (for some set $S \subset X \setminus P$) that has the spanner property on P: $\mathrm{dist}_G(u,v) \leq (1+\varepsilon)\,\mathrm{dist}(u,v)$ for all $u, v \in P$. The points in S are called *Steiner points*. In d-dimensional Euclidean space (henceforth denoted by $\mathbb{E}^d$) it is possible to obtain Steiner $(1+\varepsilon)$-spanners of size $\mathcal{O}_d(\varepsilon^{(1-d)/2}\log(1/\varepsilon)n)$ [17]. This is almost matched by a lower bound of [6], who prove that $\Omega_d(\varepsilon^{(1-d)/2}n)$ edges are required.

In this paper, we attempt to generalise Euclidean (Steiner) spanners to curved (non-Euclidean) spaces. In particular, we study the following question.

▶ **Question 1.** *What is the optimal trade-off between the stretch and the size of a (Steiner)* $(1+\varepsilon)$*-spanner in non-Euclidean spaces?*

To begin our investigation with the simplest non-Euclidean spaces, let us consider the two other geometries of constant curvature: spherical geometry,[1] representing constant positive (sectional) curvature, and hyperbolic geometry, representing constant negative (sectional) curvature. We note that spanner results apply regardless of the constant positive curvature of the spherical space, as a scaling of the metric can generate the same result for any other spherical space. The same observation holds in the hyperbolic setting. For these reasons, we work in d-dimensional spherical and hyperbolic space of sectional curvature 1 and -1, and denote them by $\mathbb{S}^d$ and $\mathbb{H}^d$, respectively.

Spherical space is naturally modelled by a d-dimensional sphere in $(d+1)$-dimensional Euclidean space, where distances are measured along the sphere. Due to its positive curvature, spherical space grows more slowly than Euclidean space, especially at larger distances (e.g., the area of a disk of radius r is $2\pi(1-\cos r)$ rather than πr^2). In contrast, hyperbolic space grows more quickly than Euclidean space (e.g., the area of a disk of radius r is now $2\pi(\cosh r - 1)$, which is $\Theta(e^r)$ for superconstant r). It is commonly visualised using

[1] Arguably, elliptic geometry is more appropriate, but less readily perceived. Our results also extend to elliptic geometry with a simple trick, which is presented in the full version.

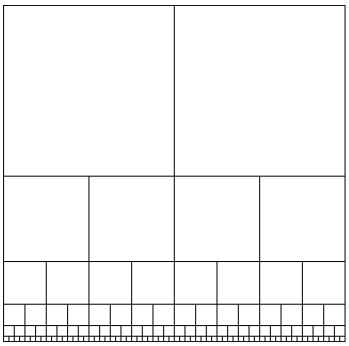


Figure 1 A patch of the binary tiling shown in the half-plane model. Note that all tiles are isometric.

models that map it to Euclidean space with heavily distorted distances, similar to e.g. the stereographic and gnomonic projection used for spheres. See for example Figure 1, which shows a tiling of the hyperbolic plane with tiles that are, importantly, isometric. Note the strong similarity to a binary tree.

One of the main motivations for studying curved spaces is that certain types of data can be much more accurately and efficiently modelled in a curved space than in (flat) Euclidean space. In particular, constant-dimensional hyperbolic space already allows for exponential growth and unbounded doubling dimension, giving good embeddings for graphs with an inherent hierarchy, such as those based on the internet [57] and social networks [62]. This has attracted the attention of, for example, the machine learning [50, 27, 12, 52], graph visualisation [46] and complex network modelling [43] communities. From a more theoretical standpoint, negative curvature is one of the simplest ways a geometric space can be non-doubling. Properly dealing with this requires new approaches that do not rely on doubling dimension and could inspire future work in other non-doubling spaces. Additionally, every closed surface can be assigned constant curvature, making especially hyperbolic geometry a useful tool in computational topology [23, 18, 22].

Both $\mathbb{S}^d$ and $\mathbb{H}^d$ are locally Euclidean spaces, meaning that a small radius ball in each space can be embedded into $\mathbb{E}^d$ with very small distortion. On a larger scale however, they have a very different behaviour. In $\mathbb{S}^d$, we can still bound the doubling dimension by $\mathcal{O}(d)$, and thus doubling metric results apply, yielding spanners of size $\varepsilon^{-\mathcal{O}(d)}n$. One would expect that the stronger Euclidean spanner and Steiner spanner constructions (of size $\mathcal{O}_d(\varepsilon^{1-d}n)$ and $\mathcal{O}_d(\varepsilon^{(1-d)/2}\log(1/\varepsilon)n)$) should be possible in $\mathbb{S}^d$, however, proving it is non-trivial: the geometric transformations to convert between these cases introduce constant multiplicative error, thus a spanner property in $\mathbb{E}^d$ does not imply the spanner property in $\mathbb{S}^d$. Note that spanners of $\mathbb{S}^d$ with the same guarantees as the Euclidean constructions would be (nearly) optimal in $\mathbb{S}^d$, as the Euclidean lower-bound constructions of Bhore and Tóth [6] can be ported to $\mathbb{S}^d$ inside a small neighbourhood. (Naturally, the Euclidean lower bounds hold in $\mathbb{H}^d$ for the same reason.)

In d-dimensional hyperbolic space ($\mathbb{H}^d$), the landscape is very different. First of all, $\mathbb{H}^d$ is not a doubling space, so doubling space results do not apply. Second, as observed by [42, 38], there do not exist t-spanners for any $t < 2$ of size less than $\binom{n}{2}$. In fact, many of the important techniques such as WSPDs and locality sensitive orderings are known to fail [38]. However, Krauthgamer and Lee [42] obtained a Steiner $(1+\varepsilon)$-spanner with $\varepsilon^{-\mathcal{O}_d(1)}n$ edges in a more general setting of visual geodesic Gromov-hyperbolic spaces. Their

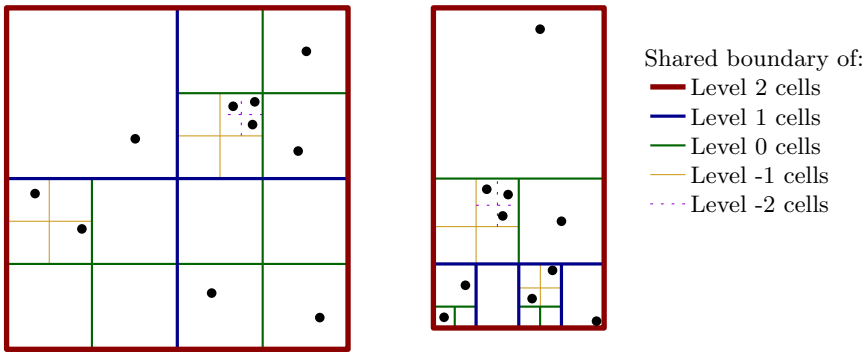


Figure 2 A Euclidean quadtree and the hyperbolic quadtree of [38].

technique is based on the observation that over shorter distances, hyperbolic space has bounded doubling dimension and thus one can use doubling space techniques, while at larger distances hyperbolic space is sufficiently "tree-like" and one can obtain additive Steiner spanners. In an *additive* (Steiner) spanner, the distance in the spanner is at most ℓ longer than the hyperbolic distance. Recently, Park and Vigneron [51] gave a Steiner spanner in $\mathbb{H}^d$ with additive error $\mathcal{O}(k \log d)$ and $2^{\mathcal{O}(d)} n\lambda_k(n)$ edges, where $\lambda_k(n)$ is an extremely slow-growing k-th row inverse Ackermann function. Finally, Kisfaludi-Bak and Van Wordragen [38] gave a dynamic Steiner $(1+\varepsilon)$-spanner in $\mathbb{H}^d$ with $\mathcal{O}(\varepsilon^{-d} \log(1/\varepsilon)n)$ edges based on the construction of a new hyperbolic quadtree. Their quadtree is shown in Figure 2. In short, it splits cells that are sufficiently small the same way as a Euclidean quadtree, while splitting larger cells into one top cell and a possibly unbounded number of bottom cells. From their Steiner spanner, [38] also obtain linear size $(2+\varepsilon)$-spanners, leaving the matter of 2-spanners open.

▶ **Question 2.** *Are there 2-spanners of size $\mathcal{O}_d(n)$ in $\mathbb{H}^d$?*

Hyperbolic spaces can exhibit very different behaviours than Euclidean spaces due to their non-doubling nature at larger scales. The tree-likeness of hyperbolic spaces has been used to obtain stronger structural and algorithmic results for several geometric problems [36, 37, 7] in $\mathbb{H}^d$ than what is possible in $\mathbb{E}^d$. It is thus natural to ask the following question.

▶ **Question 3.** *Are there (Steiner) spanners in $\mathbb{H}^d$ that have even stronger guarantees than their Euclidean counterparts for points that are pairwise distant?*

The existence of additive Steiner spanners already answers this question affirmatively, but finding the best trade-off between the additive error of a Steiner spanner and its edge count remains open.

Outside the specific hyperbolic setting, there is growing interest in obtaining spanners in other non-doubling spaces, in particular, metric spaces induced by planar, minor-free, or bounded treewidth graphs as well as polyhedral terrains or surfaces. In recent work, the technique of *tree covers* has gained momentum. A tree cover can be thought of as a set of weighted trees whose union gives a (Steiner) spanner. More precisely, set $\mathcal{T}$ is a tree cover for a point set P in metric space X if for any pair of points in P there is a tree in $\mathcal{T}$ where their distance is at most $1+\varepsilon$ times larger than their distance in X, while their distance in all trees of $\mathcal{T}$ is at least that in X. In geometric settings, the trees are typically required to be geometric graphs, i.e. any Steiner points are points in X and the edge weights match the distances in X. Chang et al. [16] gave (Steiner) tree covers in planar and treewidth t metrics, which yield Steiner $(1+\varepsilon)$-spanners of size $\mathcal{O}\big(\varepsilon^{-3} \log \frac{1}{\varepsilon} n\big)$ for planar metrics and

$2^{(t/\varepsilon)^{\mathcal{O}(t)}}n$ for treewidth-t metrics. Even more recently, Bhore et al. [5] obtained a Steiner $(1+\varepsilon)$-spanner on planar metrics and on 2-dimensional polyhedral surfaces of bounded genus with $\mathcal{O}\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\cdot\log\frac{\alpha(n)}{\varepsilon}\cdot n\right)$ edges, where $\alpha(n)$ is the inverse Ackermann function. These settings vastly generalize the finite metrics induced by point sets in $\mathbb{H}^2$ and thus the trade-offs in these settings are not directly comparable. However, the hyperbolic plane does have many of the important properties of more general planar metrics: for example, there is a natural lower bound of 2 for the stretch of (non-Steiner) spanners of subquadratic size both in planar metrics and in $\mathbb{H}^2$ [5, 38].

Limitations of previous Steiner spanner constructions. Before stating the new contribution, let us first examine why the ideas behind the existing Steiner spanner constructions in $\mathbb{H}^d$ are insufficient to get optimal sparsity. First, the Steiner spanner of Krauthgamer and Lee [42] considers a more general setting than $\mathbb{H}^d$, which among other things means they get a dependence on the local doubling dimension and consequently dependencies on d will always be of the form $\mathcal{O}(d)$. Since we care about the exact constants in the exponent of ε, this makes it unsuitable. Kisfaludi-Bak and Van Wordragen [38] place Steiner points at distance ε in a d-dimensional volume, which naturally gives a dependence ε^{-d}. It is not obvious how one can place Steiner points only at distance $\sqrt{\varepsilon}$ and along $(d-1)$-dimensional boundaries to get the optimal dependence $\varepsilon^{(1-d)/2}$. Moreover, they never have edges between two Steiner points (giving a "bipartite" Steiner spanner) and we show in Lemma 10 that such edges are necessary to get a Steiner spanner with optimal sparsity. Finally, Park and Vigneron [51] give a Steiner spanner with additive error $\mathcal{O}(\log d)$, which is good for large distances but does not help with the most challenging regime of distances between 1 and $\frac{1}{\varepsilon}$.

Our contribution. Our main contribution is the following theorem.

▶ **Theorem 1** (Main theorem). *Let $\varepsilon\in(0,\frac{1}{2}]$ and P be a set of n points in d-dimensional Euclidean, hyperbolic or spherical space. There is a Steiner $(1+\varepsilon)$-spanner for P with $\mathcal{O}_d\left(\varepsilon^{(1-d)/2}\log\frac{1}{\varepsilon}\cdot n\right)$ edges. We can maintain it dynamically, such that each point insertion or deletion takes $\mathcal{O}_d\left(\varepsilon^{(1-d)/2}\log\frac{1}{\varepsilon}\cdot\log n\right)$ time.*

Our edge count nearly matches the lower bound $\Omega\left(\varepsilon^{(1-d)/2}n\right)$ given by Bhore and Tóth [6] for Euclidean Steiner spanners, up to the factor $\log\frac{1}{\varepsilon}$. Our $\mathcal{O}_d$ notation hides a factor of $d^{\mathcal{O}(d)}$. Note that the Euclidean lower bound is also a valid lower bound in $\mathbb{S}^d$ and $\mathbb{H}^d$, as both spaces are locally Euclidean, meaning that any hard instance in $\mathbb{E}^d$ can be embedded both in $\mathbb{S}^d$ and $\mathbb{H}^d$ with arbitrarily small distortion.

Thus, we nearly settle Question 1 in the three geometries cases of constant dimension, up to the $\log 1/\varepsilon$ term. In Euclidean space our edge count matches a recent result of Chang et al. [17], however, our technique is completely different and arguably simpler. The main advantage is that our spanner can be maintained dynamically when points are added to P or deleted from P, using $\mathcal{O}_{d,\varepsilon}(\log n)$ time updates. As a result of the dynamicity and the simple structure, our spanner can readily act as a data structure for classic query problems such as dynamic approximate nearest neighbours and (dynamic approximate) bichromatic closest pair. From the Steiner spanner constructions, one can also directly get (dynamic) Steiner tree covers of size $\mathcal{O}_d\left(\varepsilon^{(1-d)/2}\log\frac{1}{\varepsilon}\right)$, which generalise the result of [17].

The main challenge in establishing Theorem 1 lies in the hyperbolic variant. While our spanner for $\mathbb{S}^d$ can be relatively easily derived from our Euclidean construction using geometric transformations, our spanner for $\mathbb{H}^d$ is substantially different. In our approach,

we use Euclidean (doubling space) ideas for small distances and tree-based ideas (constant-additive spanner) for sufficiently large distances, but in the hyperbolic setting, this leaves a gap for point pairs at distances between 1 and $\frac{1}{\varepsilon}$. This is the most challenging regime and requires mixing ideas from both approaches, as well as new ideas. As a result, we obtain a significantly smaller hyperbolic Steiner spanner compared to the state of the art, which had size $\mathcal{O}_d(\varepsilon^{-d}\log(1/\varepsilon)n)$ [38].

Note that the spanners of Theorem 1 are bipartite in case of $\mathbb{E}^d$ and $\mathbb{S}^d$, and we are also able to design bipartite Steiner spanners in $\mathbb{H}^d$ for any $d \geq 3$ with the same properties as stated in Theorem 1. Surprisingly, we prove that a bipartite Steiner spanner of size $o(n/\varepsilon)$ does not exist in $\mathbb{H}^2$. As a corollary of this lower bound, we obtain an $\Omega(n\log n)$ lower bound for the edge count of a hyperbolic (non-Steiner) 2-spanner in the plane. We prove that there exist 2-spanners in $\mathbb{H}^d$ of size $\mathcal{O}_d(n\log n)$, matching our lower bound, and settling Question 2.

▶ **Theorem 2.** *Let $P \subset \mathbb{H}^d$ be a set of n points. We can construct a 2-spanner for P with $d^{\mathcal{O}(d)} \cdot n\log n$ edges in $d^{\mathcal{O}(d)} \cdot n\log n$ time.*

By incorporating so-called *transitive closure spanners* [55] on certain trees obtained from hyperbolic quadtrees, we are also able to get Steiner spanners with ε *additive* error for any $\varepsilon > 0$ whose size is very close to linear in n: it has size $\mathcal{O}_{\varepsilon,d}(n\log\alpha(n))$, where α is the inverse Ackermann function.

▶ **Theorem 3.** *Let $\varepsilon \in (0, \frac{1}{2}]$ and P be a set of n points in $\mathbb{H}^d$. An ε-additive Steiner spanner for P with $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2}\log\frac{\alpha(n)}{\varepsilon}\cdot n$ edges can be found in $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2}\log\frac{\alpha(n)}{\varepsilon}\cdot n\log n$ time.*

While no direct lower bounds are known for additive hyperbolic spanners, this stronger variant of an additive spanner is in stark contrast with the Euclidean setting where no subquadratic-size additive (Steiner) spanners can exist: indeed, in the Euclidean case one can scale any point set, so a scale-invariant spanner construction cannot give constant additive error. In the hyperbolic setting, no scaling transformation exists, and the thin triangle property [32] allows the construction of Steiner spanners with constant additive error. In terms of Question 3, this is further evidence that pairwise distant points in $\mathbb{H}^d$ are structurally simpler than their Euclidean counterparts. It is an interesting open question whether one can improve the edge count by removing the $\log\alpha(n)$ term from the bound of Theorem 3.

Finally, we apply our spanners to two different problems. First, we observe that our spanner construction easily extends to certain closed Riemannian manifolds of constant curvature: in particular, closed constant-curvature orientable surfaces and higher-dimensional manifolds obtained as a quotient of the three base geometries using a finite isometry group.

Next, we demonstrate the usefulness of our spanners by establishing *dynamic* data structures for approximate nearest neighbours (ANN) and bichromatic closest pair. Hyperbolic ANN algorithms have already been proposed and implemented, albeit without worst-case guarantees on the running time [63, 53]. In the $(1+\varepsilon)$-approximate bichromatic closest pair problem, we are given two point sets R and B and wish to find a pair $(r,b) \in R \times B$ whose distance is at most $1+\varepsilon$ more than that of the closest pair between R and B. The dynamic version requires that the approximate bichromatic closest pair can be recalculated efficiently after point insertions and removals in R and B. Here, we improve over the result of [29] in Euclidean space, which uses $\mathcal{O}_d\big(\varepsilon^{1-d}\log(1/\varepsilon)n\big)$ space and handles updates in $\mathcal{O}_d\big(\varepsilon^{1-d}\log(1/\varepsilon)\log n\big)$ time, with a data structure that uses $\mathcal{O}_d\big(\varepsilon^{(1-d)/2}\log(1/\varepsilon)n\big)$ space and handles updates in $\mathcal{O}_d\big(\varepsilon^{(1-d)/2}\log(1/\varepsilon)\log n\big)$ time. We also improve on the result of [38] in hyperbolic space, where the factors ε^{-d} are improved to $\varepsilon^{(1-d)/2}$ in both the running

time and space usage bounds. To the best of our knowledge we give the first data structure for approximate bichromatic closest pair that is explicitly for spherical space. We obtain analogous improvements compared to the state of the art for (dynamic) ANN in hyperbolic and spherical space.

Organisation. Complete details are available in the full version; we give abridged preliminaries in Section 2 and an overview of the main ideas and constructions in Section 3. Finally, Section 4 concludes the paper by proposing some intriguing open problems.

2 Preliminaries

Throughout the paper, we will use pq to denote the shortest curve (*segment*) connecting p and q in the current geometry. In particular, this means it may not appear as a straight line segment in the used model of that geometry. Additionally, we use $|pq|$ for the length of this segment, again measured in the current geometry.

Our algorithmic results rely on compressed quadtree operations and therefore require access to floor and bitwise operations. Apart from that, they hold both in the real RAM and word RAM model. In particular, we do not need (hyperbolic) trigonometric functions despite these showing up in the distance formulas for spherical and hyperbolic space, since we only ever compare two distances between pairs of input points to each other. (In case of word RAM, we need to assume that points of $\mathbb{H}^d$ are given with coordinates in the half-space model, as we do not have the capacity to convert between different models of hyperbolic geometry.) In $\mathbb{S}^d$, any reasonable representation[2] of points can work with our algorithm. We will have the points represented in polar coordinates.

Hyperbolic geometry. In this paper we use the half-space model of hyperbolic geometry (for more details, refer to for example [11] or the textbooks [34, 61, 4]). This assigns any point $p \in \mathbb{H}^d$ coordinates $(x(p), z(p)) \in \mathbb{R}^{d-1} \times \mathbb{R}_{>0}$ in the upper half-space, which also lets us refer to the positive z-direction as "up". Important to note is that hyperbolic distances are not given by the Euclidean distances in the model, which also means that the shortest curve pq between points $p, q \in \mathbb{H}^d$ is often not a Euclidean segment. In general, pq is modelled by an arc from a Euclidean circle perpendicular to the Euclidean hyperplane $z = 0$ (in particular its centre lies at $z = 0$) and pq has hyperbolic length

$$|pq| = 2 \operatorname{arsinh} \sqrt{\frac{\|x(p) - x(q)\|^2 + (z(p) - z(q))^2}{4z(p)z(q)}},$$

but if $x(p) = x(q)$ then pq is a vertical Euclidean line segment and $|pq| = \left|\ln \frac{z(p)}{z(q)}\right|$. Note here that applying any isometry of $\mathbb{E}^{d-1}$ to $x(p)$ and $x(q)$ yields the same distance, as does multiplying all of $x(p), x(q), z(p)$ and $z(q)$ by the same positive factor. Vertical Euclidean hyperplanes are also hyperbolic hyperplanes (as with vertical segments), but horizontal ones are not. These are *horospheres*, but we will not directly use any property of horospheres other than them being distinct from hyperplanes.

[2] Note that unlike $\mathbb{H}^d$, the conversions between different representations of $\mathbb{S}^d$ and distance computations can typically be carried out with exponentially small error, which is sufficient for constructing $(1+\varepsilon)$-spanners.

Hyperbolic quadtree. Here we describe the quadtree of [38] while introducing new notation that will be useful throughout this paper. Instead of being based on hypercubes, its cells are *cube-based horoboxes*. In the half-space model, a cube-based horobox C is a Euclidean axis-parallel box with corners $(x_{\min}(C), z^{\downarrow}(C))$ and $(x_{\max}(C), z^{\uparrow}(C))$ with its *width* given by $\frac{x_{\max}(C)-x_{\min}(C)}{z^{\downarrow}(C)} = (w(C), \dots, w(C))$ and *height* $h(C) = \log \frac{z^{\uparrow}(C)}{z^{\downarrow}(C)}$. This definition ensures that any two cube-based horoboxes with the same width and height are isometric. Just like any box can tile Euclidean space, any horobox can tile hyperbolic space.

Instead of being based on the grid (the tiling of $\mathbb{E}^d$ with isometric hypercubes) like the Euclidean quadtree, the hyperbolic quadtree is based on the *binary tiling*. To construct a binary tiling, we start with the cube-based horobox C with $x_{\min}(C) = (0, \dots, 0)$, $z^{\downarrow}(C) = 1$, $w(C) = \frac{1}{\sqrt{d-1}}$ and $h(C) = 1$. Now we apply the isometry $T_{\sigma,\tau}(x, z) = \sigma \cdot (x + \tau, z)$ for all values $\tau = a/\sqrt{d-1}$ with $a \in \mathbb{Z}^{d-1}$ and $\sigma = 2^b$ with $b \in \mathbb{Z}$. This gives a tiling that covers all of $\mathbb{H}^d$ with isometric tiles; see Figure 1. Note that the two-dimensional tiling resembles a binary tree, hence the name. In general the d-dimensional tiling bears resemblance to a $(d-1)$-dimensional Euclidean quadtree, where each "layer" of the tiling at a fixed z value is a level of this quadtree.

We formalise this tree structure by thinking of the binary tiling as the infinite arborescence T^0, which has the tiles as its vertices and a directed edge from any tile to the tile directly above it. We can then also generalise this to T^ℓ for $\ell \in \mathbb{N}$. The tiling for T^ℓ is constructed by taking the cube-based horobox C with again $x_{\min}(C) = (0, \dots, 0)$ and $z^{\downarrow}(C) = 1$, but now $w(C) = 2^{2^\ell - 1}/\sqrt{d-1}$ and $h(C) = 2^\ell$. Then, we apply $T_{\sigma,\tau}$ for all values $\tau = a \cdot w(C)$ with $a \in \mathbb{Z}^{d-1}$ and $\sigma = 2^{b \cdot h(C)}$ with $b \in \mathbb{Z}$. The arborescence T^ℓ follows as before, where it is worth noting that the in-degree of vertices of T^ℓ has grown from 2^{d-1} to $2^{(d-1)\cdot 2^\ell}$.

The tiles are chosen exactly so that each tile of $T^{\ell+1}$ is the union of a tile of T^ℓ with its children. Thus, the tilings for T^ℓ can be used to define the hyperbolic quadtree for levels $\ell \geq 0$. Levels $\ell < 0$ are now constructed by splitting each cell at level $\ell + 1$ up with a Euclidean quadtree split. See Figure 2 for an illustration.

Since we will make frequent use of trees and arborescences, we also introduce some related notation for an arbitrary tree or arborescence of cells T. Given two cells $C, C' \in V(T)$, we let $\mathrm{lca}(C, C', T)$ denote their lowest common ancestor in T. Given a point $p \in \mathbb{H}^d$, we let $\mathrm{cell}(p, T)$ denote the cell of T containing p, and given a second point $q \in \mathbb{H}^d$ we use the shorthand $\mathrm{lca}(p, q, T) = \mathrm{lca}(\mathrm{cell}(p, T), \mathrm{cell}(q, T), T)$.

3 Technical overview and key ideas

All of our results are built on quadtrees. A quadtree is a simple hierarchical decomposition that is often used for geometric approximation in Euclidean space. More precisely, each of our spanners are built on $\mathcal{O}(d)$ different shifted quadtrees. It is a well-known fact that in Euclidean space, one can find $\mathcal{O}(d)$ shifts with the following key property.

(*) Any pair of points p, q is contained in a cell of diameter $|pq| \cdot \mathcal{O}(d\sqrt{d})$.

To get a set of shifted quadtrees in spherical space, we take two stereographic projections and construct shifted Euclidean quadtrees in both of these projections. For a pair of points that are not too distant, one of the stereographic projections will always give constant distortion compared to Euclidean space, and Property (*) can be achieved in one of the shifts for this projection. In hyperbolic space we use the quadtree of [38] that already comes with the required guarantee. For small enough distances, hyperbolic space also has a constant distortion compared to Euclidean space and as a result, the quadtrees share most of the properties of the Euclidean quadtree in constant-size neighbourhoods.

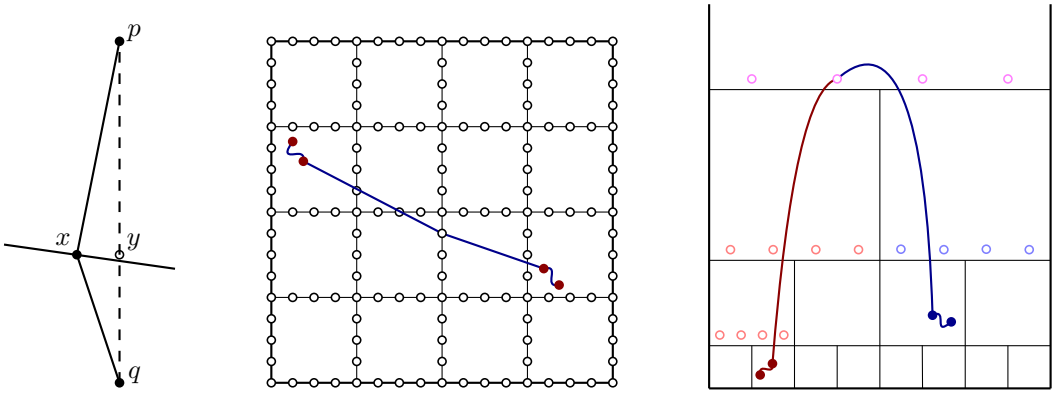


Figure 3 Left: if distance $|xy| \approx |pq|\sqrt{\varepsilon}$, then the detour from p to q through x only adds a factor $1+\varepsilon$. Middle: the Steiner spanner in $\mathbb{E}^2$. Right: the bipartite Steiner spanner in $\mathbb{H}^2$.

While constant distortion is enough to get constant-approximations, it is not enough for a $(1+\varepsilon)$-approximation. For this, we prove in Lemma 4 that a simple approximation result in Euclidean space also carries through to spherical and hyperbolic space. The following lemma is what any Euclidean Steiner spanner fundamentally relies on to get a dependency of $\varepsilon^{(1-d)/2}$ rather than ε^{1-d}. Namely, it means that for two points p, q separated by a hyperplane that has distance $\Omega(|pq|)$ from both, we can afford to go through a point x on that hyperplane with distance $\mathcal{O}(\sqrt{\varepsilon}|pq|)$ to pq; see the left-hand side of Figure 3. We generalise the lemma to spherical and small-distance hyperbolic geometry.

▶ **Lemma 4.** *Let $p, q, x \in \mathbb{X}^d$ for $\mathbb{X} \in \{\mathbb{H}, \mathbb{E}, \mathbb{S}\}$ and let y be the closest point to x on the segment pq. Assume $\varepsilon \in (0, 1]$. If $|xy| \leq \sqrt{\varepsilon}\min\{|py|, |qy|\}$, then (i) $|px| + |xq| \leq (1+\varepsilon)|pq|$ if $\mathbb{X} \in \{\mathbb{E}, \mathbb{S}\}$ and (ii) $|px| + |xq| \leq (1 + e^{\Delta}\varepsilon)|pq|$ if $\mathbb{X} = \mathbb{H}$ and $|pq| \leq \Delta$.*

With this lemma at hand, we can now give a Steiner $(1+\varepsilon)$-spanner. For any two points p and q, we consider the shifted quadtree where the cell satisfying property (*) is found. We ensure that within this cell, both p and q have some point nearby, say p' and q', respectively, such that p' and q' are both connected to a Steiner point that is chosen from a grid-like point set on a cell boundary separating p' and q'. This Steiner spanner proves Theorem 1 for $\mathbb{S}^d$, $\mathbb{E}^d$ and small-diameter point sets in $\mathbb{H}^d$.

▶ **Theorem 5.** *Let $\varepsilon \in (0, \frac{1}{2}]$ and P be a set of n points in $\mathbb{E}^d$ or $\mathbb{S}^d$, or in a quadtree cell in $\mathbb{H}^d$ of level $4 + \log\log d$. There is a Steiner $(1+\varepsilon)$-spanner for P with Steiner vertex set Q, that is bipartite on P and Q, and where each point $p \in P$ has degree $d^{\mathcal{O}(d)} \cdot \varepsilon^{(1-d)/2}\log\frac{1}{\varepsilon}$. We can maintain it dynamically such that each point insertion or deletion takes $d^{\mathcal{O}(d)} \cdot \varepsilon^{(1-d)/2}\log\frac{1}{\varepsilon} \cdot \log n$ time.*

By construction this is a *bipartite* Steiner spanner (i.e. every edge is between an input point and a Steiner point). This makes it easy to turn it into a spanner without Steiner points. For that, we take each Steiner point s and connect all input points that were connected to s to each other instead. This maintains the same bound on the stretch, but adds more edges based on the degrees of the Steiner points. In this case, the Steiner points have small degree, which leads to a total of $\mathcal{O}_d(n/\varepsilon^d)$ edges in Theorem 6.

▶ **Theorem 6.** *Let $\varepsilon \in (0, \frac{1}{2}]$ and P be a set of n points in $\mathbb{E}^d$ or $\mathbb{S}^d$, or in a quadtree cell in $\mathbb{H}^d$ of level $4 + \log\log d$. In $d^{\mathcal{O}(d)} \cdot \varepsilon^{-d} \log \frac{1}{\varepsilon} \cdot n \log n$ time, we can construct a $(1+\varepsilon)$-spanner (i.e., without Steiner points) for P with maximum degree $d^{\mathcal{O}(d)} \cdot \varepsilon^{-d} \log \frac{1}{\varepsilon}$.* [3]

Near-optimal bipartite Steiner spanner in hyperbolic space. Our first challenge here is that for super-constant hyperbolic distances the hyperbolic quadtree gets meaningfully different from a Euclidean quadtree. This was already dealt with to get a bipartite Steiner spanner in [38], but we need new insights to get near-optimal dependence on ε. The results that made the construction for small distances work are not enough, as the exponential expansion of hyperbolic space causes them to break down. In particular, Lemma 4 is false for super-constant hyperbolic distances: as the distances grow the hyperbolic Pythagorean theorem converges to the triangle inequality, which precludes us from proving something stronger for large hyperbolic distances. At medium hyperbolic distances (between 1 and $1/\varepsilon$), we give a result reminiscent of Lemma 4 to enable dependences $\varepsilon^{(1-d)/2}$ instead of ε^{1-d}.

▶ **Lemma 7.** *Let $p, q \in \mathbb{H}^d$ be separated from each other by hyperplane H that has distance at least 2 from both. For a point $x \in H$ at distance δ from pq, we get $|px| + |xq| \leq |pq| + \delta^2$.*

Note that when $|pq| = \Theta(1)$ and $\delta = \sqrt{\varepsilon}|pq|$ this is very similar to Lemma 4, while when $|pq| = \Theta(1/\varepsilon)$ and $\delta = \varepsilon|pq|$ we get the same guarantee as from the triangle inequality; the most interesting regime of the lemma is in between these two values. The hyperbolic bipartite Steiner spanner now follows from a similar construction as the one for smaller distances, with two major changes. First, we replace Lemma 4 with Lemma 7. Second, we only connect each point to its "ancestors" in the tiling (here one should think of ancestors in the binary tree), as opposed to all tiles, because in the hyperbolic quadtree there can be unboundedly many sibling tiles inside a parent cell. With this Steiner spanner we can prove Theorem 1 for all cases except for super-constant distances in $\mathbb{H}^2$.

▶ **Theorem 8.** *Let $\varepsilon \in (0, \frac{1}{2}]$ and P be a set of n points in $\mathbb{H}^d$. There is a Steiner $(1+\varepsilon)$-spanner for P with Steiner vertex set Q that is bipartite on P and Q. Each point in P has degree at most $d^{\mathcal{O}(d)} \cdot \varepsilon^{(1-d)/2} \log \frac{1}{\varepsilon}$ when $d \geq 3$ and at most $\mathcal{O}\big(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon}\big)$ when $d = 2$.*

The coming lower bound shows that, surprisingly, having a weaker result for specifically $\mathbb{H}^2$ is unavoidable unless we go for a very different, non-bipartite construction. First, Observation 9 gives a simple alternative procedure to turn a bipartite Steiner spanner into a spanner without Steiner points (as already observed by [38]). The procedure maintains the same edge bound, but in the worst case it doubles the stretch from $(1+\varepsilon)$ to $2+2\varepsilon$. As a result, we improve the $(2+\varepsilon)$-spanner of [38], but perhaps more importantly it lets us apply the coming spanner lower bound to bipartite Steiner spanners. Moreover, we will later use the same strategy to make a (non-Steiner) 2-spanner.

▶ **Observation 9** (Corollary 17 of [38]). *Given a point set P in some metric space (X, dist) and a bipartite Steiner $(1+\varepsilon)$-spanner for P with $|E|$ edges, there is also a $(2+2\varepsilon)$-spanner for P with at most $|E|$ edges.*

Lower bound. For the lower bound, we show that a $(2 + \frac{1}{\ln n})$-spanner for a set of n evenly spaced points on a circle of radius $3 \ln n$ already requires $\Omega(n \log n)$ edges. This implies three things for these point sets:

[3] Note that we do not get optimal edge count $\mathcal{O}_d\big(n/\varepsilon^{d-1}\big)$, so one can likely improve on this by adapting for example [29].

- a $(2+\varepsilon)$-spanner needs $\Omega(n/\varepsilon)$ edges,
- a bipartite Steiner $(1+\varepsilon)$-spanner needs $\Omega(n/\varepsilon)$ edges (through Observation 9), and
- a 2-spanner needs $\Omega(n \log n)$ edges.

▶ **Lemma 10.** *For any $n \geq 2$, let $\varepsilon \in [0, \frac{1}{2 \ln n}]$ and $P \subset \mathbb{H}^2$ be a set of n points equally spaced around a circle of radius $3 \ln n$. Any $(2+\varepsilon)$-spanner for P must have $\Omega(n \log n)$ edges.*

Proof sketch. First of all, the smallest and largest distance among the points differ only by a multiplicative factor close to $\frac{3}{2}$, which lets us conclude that the path between any two points p and q in the $(2+\frac{1}{\ln n})$-spanner can have at most 2 hops. We denote by m the middle vertex of the 2-hop path. Next, we observe that the location of m is highly constrained: the number of points on the circle separating m from p and q must be at most a constant times larger than the number of points separating p from q. This observation lets us partition all *pairs* of points into classes based on the number of points separating them, of the form $[51^i, 51^{i+1})$. We prove that for any pair of points, the longer edge on the path connecting them can only be shared with one disjoint pair of the same class. Thus, we require $\Theta(n)$ edges per class, and get the $\Omega(n \log n)$ lower bound from having $\Theta(\log n)$ classes. ◀

Non-bipartite Steiner spanners. To bypass the lower bound, we then work towards a non-bipartite Steiner spanner (i.e., one where Steiner points can be connected to Steiner points). The key to making this spanner work is to make use of the exponential divergence of lines in hyperbolic space: two long segments whose endpoints lie close together get exponentially close near their middle. We capture this property in Lemma 11.

▶ **Lemma 11.** *Let C be a cell of $T^{1+\log\log(1/\varepsilon)}$, let p be a point that lies in C or one of its descendant cells, and let q be a point that lies in an ancestor cell of the parent C_1 of C. Then, pq passes within distance ε of $s_C = \left(\frac{x_{\min}(C)+x_{\max}(C)}{2}, \sqrt{z^{\downarrow}(C_1) z^{\uparrow}(C_1)}\right)$.*

Directly from Lemma 11, we get Lemma 12 which gives a quasi-isometric embedding of any hyperbolic point set into a *union* of $\mathcal{O}(d)$ trees, similar to the much more involved quasi-isometric embedding of any hyperbolic metric space into a *product* of d trees [8, 9]. For our purposes, it implies a Steiner $(1+\mathcal{O}(\varepsilon))$-spanner with only $\mathcal{O}(dn)$ edges for hyperbolic point sets with minimum distance $\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}$.

▶ **Lemma 12.** *Let $\varepsilon \in (0, \frac{1}{2}]$ and P be a set of points in $\mathbb{H}^d$. There is a set of $\mathcal{O}(d)$ trees with P as the leaves whose union is a Steiner spanner for P with multiplicative distortion $1+\varepsilon$ and additive distortion $\mathcal{O}\left(\log\frac{1}{\varepsilon}\right)$. In particular, if all points in P have distance at least $\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}$ to each other, this is a Steiner $(1+\mathcal{O}(\varepsilon))$-spanner.*

To obtain a Steiner spanner for smaller distances, we need to eliminate the additive distortion. This comes from the sharp turn at the highest node on paths in these trees. We avoid this with ideas from Theorem 8. Namely, we consider the tiling $T^{4+\log\log d}$ and connect each (input and Steiner) point from the trees of Lemma 12 to $\mathcal{O}_d\left(\varepsilon^{(1-d)/2}\right)$ Steiner points placed for the $2\log\frac{1}{\varepsilon}$ cells above it. This gives more options for a gradual turn and with that finally proves that our main theorem Theorem 1 holds in all cases.

▶ **Theorem 13.** *Let $\varepsilon \in (0, \frac{1}{2}]$ and P be a set of n points in $\mathbb{H}^d$. There is a Steiner $(1+\varepsilon)$-spanner for P with $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2}\log\frac{1}{\varepsilon}\cdot n$ edges. We can maintain it dynamically, such that each point insertion or deletion takes $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2}\log\frac{1}{\varepsilon}\cdot\log n$ time.*

We can also get a Steiner spanner with *additive* error ε by using the same general idea as in Theorem 13. One of the issues here is that we accumulate some error in each hop along a spanner path, and we do not have a bound on the number of hops a spanner path might need to take. This problem can be mitigated by using the transitive closure spanner of Thorup [60] to reduce the number of hops to $\mathcal{O}(\alpha(n))$, and get the same error no matter how (hop-)distant two points are.

▶ **Theorem 3.** *Let $\varepsilon \in (0, \frac{1}{2}]$ and P be a set of n points in $\mathbb{H}^d$. An ε-additive Steiner spanner for P with $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2} \log \frac{\alpha(n)}{\varepsilon} \cdot n$ edges can be found in $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2} \log \frac{\alpha(n)}{\varepsilon} \cdot n \log n$ time.*

2-spanner. We match the lower bound of Lemma 10 with a 2-spanner that has $\mathcal{O}_d(n \log n)$ edges (in hyperbolic space of any dimension). To get the 2-spanner, we use two ways of getting a spanner from a Steiner spanner. For small distances we can get a $(1+\varepsilon)$-spanner from Theorem 6. For larger distances, we first construct a new bipartite Steiner spanner that gives constant additive error and has $\mathcal{O}_d(n \log n)$ edges. For this, we essentially take the trees of Lemma 12 for $\varepsilon = 1/d^8$, then connect each input point to $\mathcal{O}(\log n)$ nodes in each tree such that any pair of points is connected to a common node. This avoids the multiplicative distortion and makes the Steiner spanner bipartite. We then turn this into a 2-spanner with the approach of Observation 9, but we bound the stretch more precisely using the specific geometric configuration to avoid the additive error.

▶ **Theorem 2.** *Let $P \subset \mathbb{H}^d$ be a set of n points. We can construct a 2-spanner for P with $d^{\mathcal{O}(d)} \cdot n \log n$ edges in $d^{\mathcal{O}(d)} \cdot n \log n$ time.*

Dynamic data structures. We can maintain our three multiplicative Steiner spanners dynamically by using the so-called Z-order [48] or L-order [38] for each of our shifted quadtrees. These are specific orderings of the input point set based on post-order traversals of the respective quadtree, i.e., our data structure is essentially $\mathcal{O}(d)$ differently ordered lists of our n input points, as seen in locality-sensitive orderings [15, 29].

Comparing two points based on such an order is equivalent to asking which comes first if we add both to a quadtree and do a depth-first traversal of it, but these comparisons can be done in $\mathcal{O}(d)$ time without explicitly computing the quadtree. The multiplicative Steiner spanner constructions are such that for any input point, the edges connected to it purely depend on its neighbours in the Z- or L-order, which allows for efficient updates.

The results below follow directly from having dynamic bipartite Steiner spanners.

▶ **Theorem 14** (Dynamic approximate nearest neighbours)**.** *For n points in $\mathbb{S}^d$, $\mathbb{E}^d$ or $\mathbb{H}^d$ with $d \geq 3$, there is a data structure using $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2} \log \frac{1}{\varepsilon} \cdot n$ space, that can answer queries for a $(1+\varepsilon)$-approximate nearest neighbour in $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2} \log \frac{1}{\varepsilon} + \mathcal{O}\big(d^2 \log n\big)$ time and perform updates (point insertions and removals) in $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2} \log \frac{1}{\varepsilon} \cdot \log n$ time.*

▶ **Theorem 15** (Dynamic approximate bichromatic closest pairs)**.** *For n points in $\mathbb{S}^d$, $\mathbb{E}^d$ or $\mathbb{H}^d$ with $d \geq 3$, there is a data structure using $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2} \log \frac{1}{\varepsilon} \cdot n$ space, that can maintain an $(1+\varepsilon)$-approximate bichromatic closest pair while allowing updates (point insertions and removals) in $d^{\mathcal{O}(d)}\varepsilon^{(1-d)/2} \log \frac{1}{\varepsilon} \cdot \log n$ time.*

Quotient spaces. Finally, we note that we also get results for specific manifolds of the form $\mathbb{X}^d/\Gamma$ for $\mathbb{X} \in \{\mathbb{H}, \mathbb{E}, \mathbb{S}\}$, where points $p, q \in \mathbb{X}^d$ are unified if some isometry from Γ maps p to q. If Γ is finite, then we simulate this by simply replacing every point $p \in P$ by $|\Gamma|$ copies.

▶ **Corollary 16.** *For any set P of n points on a manifold isometric to $\mathbb{S}^d/\Gamma$, $\mathbb{E}^d/\Gamma$ or $\mathbb{H}^d/\Gamma$ for some finite isometry group Γ, there is a Steiner $(1+\varepsilon)$-spanner with $d^{\mathcal{O}(d)} \cdot \varepsilon^{(1-d)/2} \log \frac{1}{\varepsilon} \cdot n|\Gamma|$ edges.*

A similar approach works for constant curvature surfaces of any genus g that are closed, orientable and connected, since these are always isometric to manifolds of the form $\mathbb{X}^2/\Gamma$. Here, we use an idea from Despré, Kolbe and Teillaud [25] to represent each point in $\mathbb{X}^2/\Gamma$ by $g^{\mathcal{O}(g)}$ points in $\mathbb{X}^2$.

▶ **Theorem 17.** *Let S be a constant curvature surface of genus g that is closed, orientable and connected. For any set P of n points on S, there is a Steiner $(1+\varepsilon)$-spanner with $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}} \log \frac{1}{\varepsilon} \cdot n g^{\mathcal{O}(g)}\right)$ edges.*

4 Conclusion

We have given spanners of size $\mathcal{O}_d(\varepsilon^{(1-d)/2}n)$ in d-dimensional spaces of constant curvature, most importantly, even in the non-doubling hyperbolic setting. Our results raise several natural questions. We believe that the following are especially promising open problems for future work:

- Let $\mathcal{R}$ be a fixed d-dimensional Riemannian manifold (of non-constant curvature). Can one construct a Steiner spanner for any set of n points in $\mathcal{R}$ that has size $\widetilde{\mathcal{O}}_{\mathcal{R}}(\varepsilon^{(1-d)/2}n)$?
- Can we get a polynomial dependence on the genus g for Steiner spanners on (constant-curvature) surfaces?
- Is there a near-linear size ε-additive Steiner spanner in hyperbolic space that is dynamic? Can we remove the $\alpha(n)$ factor in its edge count?
- Can we strengthen Lemma 12, for example by getting a smaller number of trees or already getting a Steiner $(1+\varepsilon)$-spanner with $\mathcal{O}(dn)$ edges at distances $o(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$?

References

1. Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discret. Comput. Geom.*, 9:81–100, 1993. doi:10.1007/BF02189308.
2. Sunil Arya, Gautam Das, David M. Mount, Jeffrey S. Salowe, and Michiel H. M. Smid. Euclidean spanners: short, thin, and lanky. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 489–498. ACM, 1995. doi:10.1145/225058.225191.
3. MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Dániel Marx. Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5):21:1–21:37, 2011. doi:10.1145/2027216.2027219.
4. Riccardo Benedetti and Carlo Petronio. *Lectures on Hyperbolic Geometry*. Springer Science & Business Media, 1992. doi:10.1007/978-3-642-58158-8.
5. Sujoy Bhore, Balázs Keszegh, Andrey Kupavskii, Hung Le, Alexandre Louvet, Dömötör Pálvölgyi, and Csaba D. Tóth. Spanners in planar domains via Steiner spanners and non-Steiner tree covers. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 4292–4326. SIAM, 2025. doi:10.1137/1.9781611978322.145.
6. Sujoy Bhore and Csaba D. Tóth. Euclidean Steiner spanners: Light and sparse. *SIAM J. Discret. Math.*, 36(3):2411–2444, 2022. doi:10.1137/22M1502707.
7. Thomas Bläsius, Jean-Pierre von der Heydt, Sándor Kisfaludi-Bak, Marcus Wilhelm, and Geert van Wordragen. Structure and independence in hyperbolic uniform disk graphs. In *41st International Symposium on Computational Geometry, SoCG 2025, June 23-27, 2025, Kanazawa, Japan*, volume 332 of *LIPIcs*, pages 21:1–21:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SOCG.2025.21.

8 Sergei Buyalo and Viktor Schroeder. Embedding of hyperbolic spaces in the product of trees. *Geom. Dedicata*, 113:75–93, 2005. doi:10.1007/s10711-005-3124-9.

9 Sergei Buyalo and Viktor Schroeder. A product of trees as universal space for hyperbolic groups, 2005. arXiv:math/0509355.

10 Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, 1995. doi:10.1145/200836.200853.

11 James W. Cannon, William J. Floyd, Richard Kenyon, and Walter R. Parry. Hyperbolic geometry. In James W. Cannon, editor, *Non-Euclidean Geometry and Curvature*, chapter 2. American Mathematical Society, Providence, RI, 2017. Two-dimensional spaces. Vol. 3. doi:10.1090/mbk/110.

12 Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/0415740eaa4d9decbc8da001d3fd805f-Paper.pdf.

13 T.-H. Hubert Chan and Anupam Gupta. Small hop-diameter sparse spanners for doubling metrics. *Discret. Comput. Geom.*, 41(1):28–44, 2009. doi:10.1007/S00454-008-9115-5.

14 T.-H. Hubert Chan, Mingfei Li, Li Ning, and Shay Solomon. New doubling spanners: Better and simpler. *SIAM J. Comput.*, 44(1):37–53, 2015. doi:10.1137/130930984.

15 Timothy M. Chan, Sariel Har-Peled, and Mitchell Jones. On locality-sensitive orderings and their applications. *SIAM Journal on Computing*, 49(3):583–600, 2020. doi:10.1137/19M1246493.

16 Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenkovic, Shay Solomon, and Cuong Than. Covering planar metrics (and beyond): O(1) trees suffice. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 2231–2261. IEEE, 2023. doi:10.1109/FOCS57990.2023.00139.

17 Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenkovic, Shay Solomon, and Cuong Than. Optimal Euclidean tree covers. In *40th International Symposium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece*, volume 293 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SOCG.2024.37.

18 Hsien-Chih Chang and Arnaud de Mesmay. Tightening curves on surfaces monotonically with applications. *ACM Trans. Algorithms*, 18(4):36:1–36:32, 2022. doi:10.1145/3558097.

19 P Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SoCG '86, pages 169–177, New York, NY, USA, 1986. Association for Computing Machinery. doi:10.1145/10515.10534.

20 K. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 56–65, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.28402.

21 Gautam Das and Giri Narasimhan. A fast algorithm for constructing sparse Euclidean spanners. *Int. J. Comput. Geom. Appl.*, 7(4):297–315, 1997. doi:10.1142/S0218195997000193.

22 Éric Colin de Verdière, Vincent Despré, and Loïc Dubois. Untangling graphs on surfaces. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4909–4941. SIAM, 2024. doi:10.1137/1.9781611977912.175.

23 Éric Colin de Verdière and Jeff Erickson. Tightening nonsimple paths and cycles on surfaces. *SIAM J. Comput.*, 39(8):3784–3813, 2010. doi:10.1137/090761653.

24 Erik D. Demaine, MohammadTaghi Hajiaghayi, and Bojan Mohar. Approximation algorithms via contraction decomposition. *Comb.*, 30(5):533–552, 2010. doi:10.1007/S00493-010-2341-5.

25 Vincent Despré, Benedikt Kolbe, and Monique Teillaud. Representing infinite periodic hyperbolic Delaunay triangulations using finitely many Dirichlet domains. *Discret. Comput. Geom.*, 72(1):1–28, 2024. doi:10.1007/S00454-024-00653-X.

26 Arnold Filtser and Hung Le. Locality-sensitive orderings and applications to reliable spanners. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1066–1079. ACM, 2022. doi:10.1145/3519935.3520042.

27 Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5350–5360, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/dbab2adc8f9d078009ee3fa810bea142-Abstract.html.

28 Jie Gao, Leonidas J. Guibas, and An Thai Nguyen. Deformable spanners and applications. *Comput. Geom.*, 35(1-2):2–19, 2006. doi:10.1016/J.COMGEO.2005.10.001.

29 Zhimeng Gao and Sariel Har-Peled. Near-optimal Euclidean locality-sensitive orderings. *Journal of Computational Geometry*, 16(2):91–108, July 2025. doi:10.20382/jocg.v16i2a3.

30 Lee-Ad Gottlieb and Liam Roditty. Improved algorithms for fully dynamic geometric spanners and geometric routing. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 591–600. SIAM, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347148.

31 Lee-Ad Gottlieb and Liam Roditty. An optimal dynamic spanner for doubling metric spaces. In *Algorithms - ESA 2008, 16th Annual European Symposium, Karlsruhe, Germany, September 15-17, 2008. Proceedings*, volume 5193 of *Lecture Notes in Computer Science*, pages 478–489. Springer, 2008. doi:10.1007/978-3-540-87744-8_40.

32 M. Gromov. *Hyperbolic Groups*, pages 75–263. Springer New York, New York, NY, 1987. doi:10.1007/978-1-4613-9586-7_3.

33 Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006. doi:10.1137/S0097539704446281.

34 Birger Iversen. *Hyperbolic Geometry*, volume 25 of *London Mathematical Society Student Texts*. Cambridge University Press, 1992. doi:10.1017/CBO9780511569333.

35 J. Mark Keil. Approximating the complete Euclidean graph. In *SWAT 88*, pages 208–213, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg. doi:10.1007/3-540-19487-8_23.

36 Sándor Kisfaludi-Bak. Hyperbolic intersection graphs and (quasi)-polynomial time. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1621–1638. SIAM, 2020. doi:10.1137/1.9781611975994.100.

37 Sándor Kisfaludi-Bak. A quasi-polynomial algorithm for well-spaced hyperbolic TSP. In *36th International Symposium on Computational Geometry, SoCG 2020, June 23-26, 2020, Zürich, Switzerland*, volume 164 of *LIPIcs*, pages 55:1–55:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.SOCG.2020.55.

38 Sándor Kisfaludi-Bak and Geert van Wordragen. A quadtree, a Steiner spanner, and approximate nearest neighbours in hyperbolic space. *Journal of Computational Geometry*, 16(2):145–174, 2025. doi:10.20382/jocg.v16i2a5.

39 Sándor Kisfaludi-Bak, Jesper Nederlof, and Karol Węgrzycki. A gap-ETH-tight approximation scheme for Euclidean TSP. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 351–362, 2022. doi:10.1109/FOCS52979.2021.00043.

40 Sándor Kisfaludi-Bak and Geert van Wordragen. Near-optimal dynamic steiner spanners for constant-curvature spaces, 2025. doi:10.48550/arXiv.2509.01443.

41 Philip N. Klein. A subset spanner for planar graphs, with application to subset TSP. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 749–756. ACM, 2006. doi:10.1145/1132516.1132620.

42 Robert Krauthgamer and James R. Lee. Algorithms on negatively curved spaces. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pages 119–132. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.9.

43 Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010. doi:10.1103/PhysRevE.82.036106.

44 An La and Hung Le. Dynamic locality sensitive orderings in doubling metrics. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 1178–1189. ACM, 2025. doi:10.1145/3717823.3718209.

45 An La, Hung Le, Shay Solomon, Cuong Than, Vinayak, Shuang Yang, and Tianyi Zhang. A tight lower bound for doubling spanners, 2025. doi:10.48550/arXiv.2508.11555.

46 John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, 1995. doi:10.1145/223904.223956.

47 Hung Le and Shay Solomon. Truly optimal Euclidean spanners. *SIAM Journal on Computing*, pages FOCS19-135–FOCS19-199, 2022. doi:10.1137/20M1317906.

48 Guy M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical Report Morton1966, International Business Machines Company New York, 1966. URL: https://dominoweb.draco.res.ibm.com/0dabf9473b9c86d48525779800566a39.html.

49 Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, 2007. doi:10.1017/CBO9780511546884.

50 Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3776–3785. PMLR, 2018. URL: http://proceedings.mlr.press/v80/nickel18a.html.

51 Eunku Park and Antoine Vigneron. Embeddings and near-neighbor searching with constant additive error for hyperbolic spaces. *Computational Geometry*, 126:102150, 2025. doi:10.1016/j.comgeo.2024.102150.

52 Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. Hyperbolic deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10023–10044, 2022. doi:10.1109/TPAMI.2021.3136921.

53 Liudmila Prokhorenkova, Dmitry Baranchuk, Nikolay Bogachev, Yury Demidovich, and Alexander Kolpakov. Graph-based nearest neighbor search in hyperbolic spaces. In *International Conference on Learning Representations*, 2022. URL: https://openreview.net/forum?id=USIgIY6TNDe.

54 Satish Rao and Warren D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 540–550. ACM, 1998. doi:10.1145/276698.276868.

55 Sofya Raskhodnikova. Transitive-closure spanners: A survey. In *Property Testing - Current Research and Surveys*, volume 6390 of *Lecture Notes in Computer Science*, pages 167–196. Springer, 2010. doi:10.1007/978-3-642-16367-8_10.

56 Liam Roditty. Fully dynamic geometric spanners. *Algorithmica*, 62(3-4):1073–1087, 2012. doi:10.1007/S00453-011-9504-7.

57 Y. Shavitt and T. Tankel. On the curvature of the internet and its usage for overlay construction and distance estimation. In *IEEE INFOCOM 2004*, volume 1, page 384, 2004. doi:10.1109/INFCOM.2004.1354510.

58 Michiel Smid. The weak gap property in metric spaces of bounded doubling dimension. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, volume 5760 of *Lecture Notes in Computer Science*, pages 275–289. Springer, 2009. doi:10.1007/978-3-642-03456-5_19.

59 Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 281–290. ACM, 2004. doi:10.1145/1007352.1007399.

60 Mikkel Thorup. Parallel shortcutting of rooted trees. *J. Algorithms*, 23(1):139–159, 1997. doi:10.1006/JAGM.1996.0829.

61 William P. Thurston. *Three-Dimensional Geometry and Topology, Volume 1*. Princeton University Press, 1997. doi:10.1515/9781400865321.

62 Kevin Verbeek and Subhash Suri. Metric embedding, hyperbolic space, and social networks. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 501–510, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2582112.2582139.

63 Xian Wu and Moses Charikar. Nearest neighbor search for hyperbolic embeddings, 2020. arXiv:2009.00836.

Fast Nearest Neighbor Search for ℓ_p Metrics

Robert Krauthgamer
The Harry Weinrebe Professorial Chair of Computer Science, Weizmann Institute of Science, Rehovot, Israel

Nir Petruschka
Weizmann Institute of Science, Rehovot, Israel

Abstract

The Nearest Neighbor Search (NNS) problem asks to design a data structure that preprocesses an n-point dataset X lying in a metric space $\mathcal{M}$, so that given a query point $q \in \mathcal{M}$, one can quickly return a point of X minimizing the distance to q. The efficiency of such a data structure is evaluated primarily by the amount of space it uses and the time required to answer a query. We focus on the fast query-time regime, which is crucial for modern large-scale applications, where datasets are massive and queries must be processed online, and is often modeled by query time $\mathrm{poly}(d \log n)$ when $\mathcal{M}$ is a d-dimensional normed space. Our main result is such a randomized data structure for NNS in ℓ_p^d spaces, $p > 2$, that achieves $p^{O(1)+\log\log p}$ approximation with fast query time and $\mathrm{poly}(dn)$ space. Our data structure improves, or is incomparable to, the state-of-the-art for the fast query-time regime from [Bartal and Gottlieb, TCS 2019] and [Krauthgamer, Petruschka and Sapir, FOCS 2025].

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Nearest neighbor algorithms

Keywords and phrases Nearest neighbor search, metric embeddings, ℓ_p norm

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.66

Related Version *arXiv Version*: `https://arxiv.org/abs/2603.21148`

Funding *Robert Krauthgamer*: Work partially supported by the Israel Science Foundation grant #1336/23.

Acknowledgements We thank Shay Sapir and Asaf Petruschka for helpful discussions.

1 Introduction

The Nearest Neighbor Search (NNS) problem asks to design a data structure (also called a scheme) that preprocesses an n-point dataset X lying in a metric space $\mathcal{M}$, so that given a query point $q \in \mathcal{M}$, one can quickly return a point of X minimizing the distance to q (or approximately minimizing it in the approximate version). The efficiency of such a data structure is evaluated primarily by the amount of space it uses and the time required to answer a query. The preprocessing time is a secondary measure and is usually comparable to the space usage. Because of its central role in areas such as machine learning, data analysis, and information retrieval, NNS has been the subject of extensive research, both practical and theoretical (see, e.g., the surveys [1, 6]).

It is well known that approximate NNS can be reduced to solving $\mathrm{polylog}(n)$ instances of the approximate *near* neighbor problem [18]. For this reason, we restrict attention to the latter.

© Robert Krauthgamer and Nir Petruschka;
licensed under Creative Commons License CC-BY 4.0

▶ **Definition 1.1.** *The Approximate Near Neighbor problem for a metric space $(\mathcal{M}, d_{\mathcal{M}})$ and parameters $c \geq 1$, $r > 0$, abbreviated (c, r)-ANN, is the following. Design a data structure that preprocesses an n-point subset $X \subseteq \mathcal{M}$, so that given a query $q \in \mathcal{M}$ with $d_{\mathcal{M}}(q, X) \leq r$,*[1] *it reports $x \in X$ such that*

$$d_{\mathcal{M}}(q, x) \leq cr.$$

In a randomized data structure, the reported $x \in X$ satisfies this with probability at least $2/3$.

We focus on the fast query-time regime, which is crucial for modern large-scale applications where datasets are massive and queries must be processed online, and is often modeled by query time $\mathrm{poly}(d \log n)$ when $\mathcal{M}$ is a d-dimensional normed space. We study the metric spaces ℓ_p^d, which refer to $\mathbb{R}^d$ with distances defined by the ℓ_p norm, and we may omit the dimension d when it is clear from the context. In ℓ_p spaces, ANN in this regime is well understood for $1 \leq p \leq 2$ [19, 25, 18] and for $p = \infty$ [20, 4, 21]. For $2 < p < \infty$, the situation is less clear: there exists a handful of data structures, each suitable for a different range of p, as detailed in Table 1. We present a new scheme for ANN in ℓ_p, $p > 2$, with fast query time, that offers an improved tradeoff between approximation and space, as follows.

▶ **Theorem 1.2.** *Let $p > 2$, $d \geq 1$. Then for every $r > 0$, there is a randomized data structure for (c, r)-ANN in ℓ_p^d, where $c = p^{O(1)+\log\log p}$, that has query time $\mathrm{poly}(d \log n)$ and preprocessing (both space and time) $\mathrm{poly}(dn)$.*[2]

Studying ANN in ℓ_p, $p > 2$, is important both practically and theoretically. Real-world data and applications may motivate norms that emphasize outliers, e.g., for anomaly detection, or alter the presence of "hubs", e.g., to affect classification; such data structures were indeed used for time-series classification [28], see [14] for additional references. From a theoretical perspective, the geometry of ℓ_p spaces undermines existing algorithmic techniques and requires developing new ones. A key challenge is to bridge between $p = 2$ and $p = \infty$. In the fast query-time regime, this means interpolating between the classical $(O(1), r)$-ANN in ℓ_2 [18] and the $(O(\log\log d), r)$-ANN in ℓ_∞ [20], which both use only $\mathrm{poly}(dn)$ space. It is natural to conjecture that $2 < p < \infty$ exhibits an interpolation between these two guarantees, and since ℓ_∞^d is $O(1)$-equivalent to $\ell_{\log d}^d$ by Hölder's inequality, this interpolated data structure is conjectured to achieve $O(\log p)$ approximation using $\mathrm{poly}(nd)$ space. The first step towards this conjecture, in [2, 3], devised a reduction from ℓ_p to ℓ_∞, and obtained a data structure with $O(\log^{1/p} n \log\log d)$-approximation, which is mainly suited for large values of p. A nontrivial reduction, devised in [14], reduced ℓ_p to ℓ_2 and obtains $2^{O(p)}$-approximation, which is a major improvement for small values of p, although it is doubly-exponentially worse than the conjecture. A more sophisticated reduction, that was devised recently in [23], achieves approximation $\mathrm{poly}(p)$, which is an exponential improvement. However, it goes through multiple intermediate ℓ_t spaces ($2 \leq t < p$) via a recursive argument that increases the space to $n^{O(\log p)}$, much higher than conjectured. Our Theorem 1.2 essentially completes the improvement of [23], by decreasing the space complexity back to the conjectured $\mathrm{poly}(dn)$, albeit slightly increasing the approximation to $p^{\log\log p}$. In particular, it resolves a question posed in [23], of whether the recursion can avoid this higher space complexity.

The proof of Theorem 1.2 is based on a simple yet powerful enhancement of the known NNS schemes from [14, 23], which utilizes classical results from [11, 12] about sparse covers. We provide an overview of the algorithms of [14, 23] in Section 3, along with an intuitive explanation of our approach at the beginning of Section 4.

---

[1] If $d_{\mathcal{M}}(q, X) > r$, it may report anything, where as usual, $d_{\mathcal{M}}(q, X) := \min_{x^* \in X} d_{\mathcal{M}}(x^*, q)$.

[2] The time and space bounds are independent of p.

Table 1 Known data structures for ANN in ℓ_p, $p > 2$, in the fast query time regime.

Approximation	Space	Reference
$O(\log^{1/p} n \log\log d)$	$\text{poly}(nd)$	[2, 3]
$2^{O((\log d)^{2/3}(\log\log d)^{1/3})}$	$\text{poly}(nd)$	[8]
$2^{O(p)}$	$\text{poly}(nd)$	[14]
$p^{O(1)}$	$n^{O(\log p)}$	[23]
c	$n^{O(p/c)\cdot\log\log n}$	[13]
$p^{O(1)+\log\log p}$	$\text{poly}(nd)$	Theorem 1.2

Related Work

Many of the existing results on approximate nearest neighbor search in ℓ_p spaces focus on the case $1 \le p \le 2$ [25, 19, 18, 17, 5, 3, 27]. In this setting, $O(1)$-approximation can be achieved with $\text{poly}(dn)$ preprocessing (space and time), and query time polynomial in $d \log n$ [25, 18].

In recent years, significant progress has been made for the case $p \in (2, \infty)$, and current results can be broadly divided into three regimes. The first regime consists of data structures that achieve moderate approximation and query time using near-linear space [3, 7, 8, 9, 24, 10]. The second regime has small approximation factor, say $O(1)$ or even $1 + \varepsilon$, in which case both the query time and the preprocessing requirements (space and time) are typically very large [14, 13]. The third regime, which is the focus of our work, has fast query time, namely, $\text{poly}(d \log n)$, and existing results either achieve $2^{O(p)}$-approximation with $\text{poly}(dn)$ space [14], or better approximation $o(2^p)$ at the cost of a much bigger $n^{\omega(1)}$ space [13, 23]. Our result is the first to obtain both $o(2^p)$ approximation and $\text{poly}(dn)$ space. Table 1 provides a comparison of all known data structures in this regime.

We point out that the recursive embedding technique from [23], which is used here, has been applied successfully also to other problems involving ℓ_p spaces, such as the construction of Lipschitz decompositions [22, 23, 26], geometric spanners [22, 23], and low-distortion embeddings [23, 26].

2 Preliminaries

Given a metric space $\mathcal{M} = (X, d_{\mathcal{M}})$, we denote by $B_{d_{\mathcal{M}}}(x, r) := \{y \in X : d_{\mathcal{M}}(x, y) \le r\}$ the ball of radius $r > 0$ centered at a point $x \in \mathcal{M}$.

For every $p, q \in [1, \infty)$, the Mazur map $M_{p,q} : \ell_p^d \to \ell_q^d$ is computed by taking, in each coordinate, the absolute value raised to power p/q, but keeping the original sign. Our algorithm crucially relies on the following property of this map.

▶ **Theorem 2.1** ([15, 14]). *Let $1 \le q < p < \infty$ and $C_0 > 0$, and let M be the Mazur map $M_{p,q}$ scaled down by factor $\frac{p}{q} C_0^{p/q-1}$. Then for all $x, y \in \ell_p^d$ such that $x, y \in B_{\ell_p}(0, C_0)$,*

$$\tfrac{q}{p}(2C_0)^{1-p/q}||x - y||_p^{p/q} \le ||M(x) - M(y)||_q \le ||x - y||_p.$$

3 Overview of Algorithms from [14, 23]

In this section, we review the algorithms of [14] and [23] for ANN in ℓ_p, $p > 2$. For the rest of the section, fix a dataset $X \subset \ell_p^d$ with $|X| = n$ for some $p > 2$.

3.1 $(2^{O(p)}, r)$-ANN with $\mathrm{poly}(dn)$ Space [14]

In the preprocessing stage, consider a set of $k = \frac{p}{2} \cdot O(\log\log d)$ possible approximation factors $\hat{\mathcal{C}} = \{\hat{c}_i\}_{i=0}^{k}$, where $\mathrm{poly}(d) = \hat{c}_0 \geq \hat{c}_1 \geq \cdots \geq \hat{c}_k = 2^{O(p)}$. First, compute for X an initial NNS data structure A_{init} using [16], which provides approximation $\hat{c}_0 = \mathrm{poly}(d)$ using query time $\mathrm{poly}(d \log n)$ and space $\mathrm{poly}(d)\tilde{O}(n)$.[3] Then, for every data point $x \in X$ and every approximation $\hat{c} \in \hat{\mathcal{C}}$, compute a (scaled) Mazur map $M_{x,\hat{c}} : \ell_p^d \to \ell_2^d$ for the points in the set $B_{\ell_p}(0, \hat{c}r) \cap (X - x)$. Finally, compute for the image points in ℓ_2^d the $(2, r)$-ANN data structure $A_{x,\hat{c}}$ from [18], which uses $\mathrm{poly}(d \log n)$ query time and $\mathrm{poly}(dn)$ space. We have this data structure for each point $x \in X$ and each approximation factor $\hat{c} \in \hat{\mathcal{C}}$, and clearly $|B_{\ell_p}(x, \hat{c}r)| \leq n$, hence the total space requirement is $O(p \cdot \log\log d)n \cdot \mathrm{poly}(dn) = \mathrm{poly}(dn)$.

At query time, given a query point $q \in \ell_p^d$, find a $\hat{c}_0$-approximate solution x_0 using A_{init}. The crucial observation is that since the Mazur map ensures a distortion that depends on the diameter of the point set (Theorem 2.1), the answer from $A_{x_0 \cdot \hat{c}_0}$ is a $\hat{c}_1$-approximate solution x_1. Applying this procedure iteratively, the approximation factor decreases even faster than geometrically, roughly as $\hat{c}_i = \hat{c}_{i-1}^{1-2/p}$. Hence, after $k = O(p \log\log c_0)$ iterations we obtain an approximate solution x_k with $\hat{c}_k = 2^{O(p)}$, where the approximation factor does not improve further.

3.2 $(\mathrm{poly}(p), r)$-ANN with $\mathrm{poly}(d)n^{O(\log p)}$ Space [23]

In [23], the image space of the Mazur map is changed from ℓ_2 to ℓ_t for general $1 \leq t < p$. Generalizing the results from [14], a (c_t, r)-ANN data structure in ℓ_t^d with query time $Q(n)$ and space $S(n)$ is used to construct a $(2^{O(p/t)} \cdot c_t, r)$-ANN data structure for ℓ_p^d with query time $O(d) + \frac{p}{t}O(\log\log d)Q(n)$ and space $\frac{p}{t}O(\log\log d)n \cdot S(n)$. Using the above result with $t = p/2$, and applying it recursively to decrease p to 2 (which is actually a double recursion, because we also iterate over the $\hat{c}_i$'s), yields a $(\mathrm{poly}(p), r)$-ANN data structure with query time $\mathrm{poly}(d \log n)$. The caveat is that every application of the recursive step multiplies the space of the data structure by factor n, which yields a data structure with space $\mathrm{poly}(d)n^{O(\log p)}$.

4 $(2^{\tilde{O}(\log p)}, r)$-ANN with $\mathrm{poly}(dn)$ Space

In this section, we give the proof of Theorem 1.2. We first explain the intuition, and for simplicity we restrict this discussion to reducing the space requirement of [14]; reducing the space requirements of [23] is similar in spirit, although more technical.

Revisiting Section 3.1, the final solution x_k is obtained by finding iteratively a sequence of intermediate solutions $x_0, x_1, \ldots, x_{k-1}$. Each iteration $i < k$ makes progress by finding a point x_i and restricting the search region to $B_{\ell_p}(x_i, \hat{c}_i r)$, which has bounded diameter, and thus applying a Mazur map on this region has distortion guarantees. It follows that querying the data structure $A_{x_i,\hat{c}_i}$ (computed over $B_{\ell_p}(x_i, \hat{c}_i r) \cap X$) finds a point x_{i+1} and we can restrict the search region even further, to diameter $\hat{c}_{i+1} r$.

The preprocessing phase prepares for the possibility that each point $x \in X$ will serve (at query time) the $\hat{c}_i$-approximate solution, i.e., the search region will be restricted to $B_{\ell_p}(x, \hat{c}_i r)$. To make progress and restrict the search region even further, a data structure $A_{x,\hat{c}_i}$ is constructed for (the points in) this region. Our key idea in Theorem 1.2 is that, rather

[3] Throughout, the notation $\tilde{O}(f)$ hides factors that are logarithmic in f.

than preparing a *separate* data structure for each search region, the algorithm constructs one *global* collection of data structures that together cover all the possible search regions. For every $\hat{c}_i$, the algorithm constructs a set of ANN data structures computed on a collection of subsets $\mathcal{S} \subseteq 2^X$, such that for every point $x \in X$ there is some $S \in \mathcal{S}$ that contains the search region $B_{\ell_p}(x, \hat{c}_i r) \cap X$. In addition, every $S \in \mathcal{S}$ has diameter at most $\beta \hat{c}_i r$ for some $\beta > 1$. We also want the total number of points in $\mathcal{S}$ (counting repetitions) to be small. The preprocessing algorithm simply stores for every $x \in X$ a reference to a set $S_x \in \mathcal{S}$ with $B_{\ell_p}(x, \hat{c}_i r) \cap X \subseteq S_x$, and at query time, if x serves as a $\hat{c}_i$-approximate solution, the algorithm queries the ANN data structure constructed for S_x. Since S_x has a diameter at most $\beta \hat{c}_i r$, this will still cause the search region's diameter to shrink in the next iteration (although by a slightly smaller factor). Since the total number of points in $\mathcal{S}$ is small, the total memory used by all the ANN data structures will be small too.

It remains to show that the preprocessing phase can indeed find efficiently a collection of subsets of X with the above properties. Fortunately, this was shown to be possible in [11, 12], and has become a fundamental algorithmic tool with numerous applications in distributed computing, network design, routing, graph algorithms, and metric embeddings.

▶ **Definition 4.1** (Sparse Neighborhood Cover [11, 12]). *A (β, r)-sparse cover of a metric space $\mathcal{M} = (X, d_{\mathcal{M}})$ is a collection of subsets (called clusters) $\mathcal{S} \subseteq 2^X$, each of diameter at most βr, such that for every $x \in X$ there exists $S \in \mathcal{S}$ with $B_{d_{\mathcal{M}}}(x, r) \subseteq S$. The total number of points $\sum_{S \in \mathcal{S}} |S|$ is called the* sparsity *of $\mathcal{S}$.*

▶ **Theorem 4.2** ([12]). *There is an algorithm that, given an n-point metric space $\mathcal{M}$ and parameters $\beta > 1$ and $r > 0$, outputs a (β, r)-sparse cover of $\mathcal{M}$ of sparsity $O(n^{1+1/\beta})$, and runs in $O(\beta n^{2+2/\beta})$ time.*

We are now ready to prove Theorem 1.2, largely following the proof structure of [23, Theorem 1.8].

Proof of Theorem 1.2. Let $X \subset \ell_p^d$ be an n-point dataset for some $p \in (2, \infty)$. For clarity of exposition, we assume that p is a power of 2, which can be easily resolved, see [23, Theorem 1.2]. Also, by an application of Hölder's inequality, we may assume that $p \leq \log d$.

We construct the ANN scheme using a doubly-recursive procedure. The first recursion assumes access to an ANN scheme for ℓ_p^d that achieves approximation factor c_{base}, and provides a new ANN scheme for ℓ_p^d that achieves improved (smaller) approximation c_{new}. This step crucially relies on access to yet another ANN data structure, for ℓ_t^d, for $t = p/2$, that is actually constructed by the same method. This leads to a second recursion, of constructing ANN data structures for intermediate spaces $\ell_p^d, \ell_{p/2}^d, \ldots, \ell_2^d$, where the space ℓ_2^d is known to have ANN data structures with $O(1)$ approximation.

We next describe the first recursion, i.e., how to construct an improved (c_{new}, r)-ANN scheme for ℓ_p^d given a (c_{base}, r)-ANN scheme for ℓ_p^d and a (c_t, r)-ANN scheme for ℓ_t^d, where $t < p$. In the preprocessing phase, use Theorem 4.2 to construct for X a $(\beta, 2c_{\text{base}} r)$-cover $\mathcal{S}$ with sparsity $\tilde{O}(n^{1+1/\beta})$ for $\beta = \log p$. During the construction of $\mathcal{S}$, store for every $x \in X$ a reference to a set $S_x \in \mathcal{S}$ that "covers" it, i.e., $B_{\ell_p}(x, 2c_{\text{base}} r) \cap X \subseteq S_x$, which is guaranteed to exist in a sparse cover. In addition, for every $S \in \mathcal{S}$ designate (arbitrarily) a center point $y \in S$, apply a Mazur map $M^y : \ell_p^d \to \ell_t^d$ scaled down by factor $\frac{p}{t} \cdot (2\beta c_{\text{base}} r)^{p/t-1}$ on $B_{\ell_p}(0, 2\beta c_{\text{base}} r) \cap (X - y)$, and construct for these image points a (c_t, r)-ANN scheme A_S. Finally, construct a (c_{base}, r)-ANN scheme A_{base} for X, and amplify the success probabilities of both data structures to $5/6$ by the standard method of independent repetitions. Given a query $q \in \ell_p^d$ that is guaranteed to have $x^* \in X$ with $\|x^* - q\|_p \leq r$, query A_{base}

for the point q and obtain an answer $x_{\text{base}} \in X$. Then find its cluster $S_{x_{\text{base}}}$ and this cluster's designated center y, query $A_{S_{x_{\text{base}}}}$ for the point $M^y(q-y) \in \ell_t^d$, and use its answer $M^y(z_{\text{out}} - y) \in M^y(X-y)$ to output the corresponding $z_{\text{out}} \in X$.

The next claim is analogous to [23, Claim 4.1], and the main difference is using the sparse cover.

▷ Claim 4.3. With probability at least 2/3, we have $\|z_{\text{out}} - q\|_p \le c_{\text{new}} r$, where $c_{\text{new}} = (\frac{p}{t})^{t/p}\, c_t^{t/p}\, (4\beta c_{\text{base}})^{1-t/p}$.

Proof. With probability at least $\frac{5}{6}$, the data structure A_{base} outputs a point x_{base} with $\|x_{\text{base}} - q\|_p \le c_{\text{base}} r$. Let $S_{x_{\text{base}}}$ be the set in the cover referenced by x_{base}, and let $y \in S_{x_{\text{base}}}$ be its designated center point. Since

$$\|x^* - x_{\text{base}}\|_p \le \|x^* - q\|_p + \|q - x_{\text{base}}\|_p \le 2c_{\text{base}} r,$$

we get that $x^* \in B_{\ell_p}(x_{\text{base}}, 2c_{\text{base}} r) \cap X \subseteq S_{x_{\text{base}}}$. Observe that by Theorem 2.1, $\|M^y(x^*) - M^y(q)\|_t \le r$, and thus with probability at least $\frac{5}{6}$, querying $A_{S_{x_{\text{base}}}}$ finds a point $M^y(z_{\text{out}}) \in M^y(S_{x_{\text{base}}}) \subseteq M^y(X)$ with $\|M^y(z_{\text{out}}) - M^y(q)\|_t \le c_t r$. Applying a union bound, we see that with probability at least 2/3, both events hold. In this case, we have by Theorem 2.1, that

$$\frac{t}{p} \cdot (4\beta c_{\text{base}} r)^{1-p/t} \cdot \|z_{\text{out}} - q\|_p^{p/t} \le \|M^y(z_{\text{out}}) - M^y(q)\|_t \le c_t r,$$

and by rearranging, we obtain $\|z_{\text{out}} - q\|_p \le (\frac{p}{t})^{t/p}\, c_t^{t/p}\, (4\beta c_{\text{base}})^{1-t/p} r = c_{\text{new}} r$. ◁

Denote by $\hat{c}_i$ the approximation of the ANN scheme obtained by i applications of Claim 4.3, where the initial ANN scheme is the one from [16], with approximation $\hat{c}_0 = \text{poly}(d)$. We also denote by c_t the approximation of an ANN scheme for ℓ_t, that is constructed by the same method (i.e., recursively), except that for ℓ_2^d we use a $(2,r)$-ANN scheme from [18] with $\text{poly}(d \log n)$ query time and $\text{poly}(dn)$ space and preprocessing time. Using Claim 4.3 with $t = p/2$, and furthermore applying this recursively $k = \lceil \log(\log \hat{c}_0) \rceil = O(\log \log d)$ times, we obtain

$$\hat{c}_k \le \sqrt{8\beta c_{p/2} \hat{c}_{k-1}} \le \sqrt{8\beta c_{p/2} \sqrt{8\beta c_{p/2} \hat{c}_{k-2}}} \le \ldots \le 8\beta c_{p/2} \cdot \hat{c}_0^{1/2^k} \le 16\beta c_{p/2},$$

i.e., this scheme has approximation $c_p = \hat{c}_k \le 16\beta c_{p/2}$. We can amplify the success probability of this scheme to $1 - \frac{1}{3\log p}$ by the standard method of $O(\log \log p) = O(\log \log \log d)$ independent repetitions. Now by recursion over p for $\log p$ levels, we get that

$$c_p \le (16\beta)^{\log p} = (16 \log p)^{\log p} = p^{4 + \log \log p},$$

and the overall success probability is at least $\frac{2}{3}$ by a union bound.

We are left to analyze the query time of the algorithm, and its space and preprocessing time. Each level of the second recursion makes a total of $k \cdot O(\log \log p) = \tilde{O}(\log \log d)$ calls to an ANN scheme for ℓ_t, for different intermediate values of t. Since the $(2,r)$-ANN for ℓ_2 from [18] has query time $\text{poly}(d \log n)$, and recalling that $p \le \log d$, the overall query time is $\tilde{O}(\log \log d)^{\log p} \cdot \text{poly}(d \log n) = \text{poly}(d \log n)$.

To analyze the space and preprocessing time, we prove the following claim.

▷ Claim 4.4. There exists an absolute constant $D > 2$ such that when the data structure for ℓ_t^d, $t = 2^i$, is computed on m points, it uses total space and preprocessing time $\tilde{O}(\log \log d)^i\, \text{poly}(d) O(m)^{D + i/\log p}$.

Proof. We only analyze the space usage of the data structure; the analysis of the preprocessing time follows similarly, as it takes $O(d)$ time to compute a Mazur map and $O(\log p \cdot m^{2(1+1/\log p)}) \leq O(\log\log d \cdot m^{2(1+1/\log p)})$ time to compute a sparse cover.

The proof proceeds by induction on $i \geq 0$. For $i = 0$, the claim follows because the $(2, r)$-ANN from [18], when computed on m points, uses at most $\mathrm{poly}(d)m^D$ space for some absolute constant $D > 1$. Now, assume the claim holds for $i - 1 \geq 0$. The ANN scheme at level i of the recursion consists of two types of ANN schemes. The first type is an ANN scheme from [16] computed on all m points, which uses $\mathrm{poly}(d)\tilde{O}(m) \leq \mathrm{poly}(d)O(m)^{D(1+1/\log p)}$ space. The second type are multiple ANN schemes at level $i - 1$ that are computed on different subsets of the m points. For every $j = 0, 1, \ldots, k = \tilde{O}(\log\log d)$, let $\mathcal{S}^j$ be the sparse cover of sparsity $O(m^{1+1/\log p})$ computed for the points at the j-th level of the first recursion. For every level j of the first recursion and cluster $S \in \mathcal{S}^j$, the algorithm computes a data structure of level $i - 1$ on S. By the induction hypothesis, the space of this data structure is bounded by $\tilde{O}(\log\log d)^i \,\mathrm{poly}(d)O(m)^{D+i/\log p}$. Observe that the function $f : x \mapsto \tilde{O}(\log\log d)^i \,\mathrm{poly}(d)x^{D+i/\log p}$ satisfies that $f(a) + f(b) \leq f(a + b)$ for all $a, b \geq 1$, and, since every data structure is constructed over some $S \in \mathcal{S}^j$ containing at most m points, we can bound the contribution of every point $p \in S$ to the total space usage of the data structure by $\frac{f(m)}{m}$. Since for every j the sparsity of $\mathcal{S}^j$ is at most $O(m^{1+1/\log p})$, the total space usage of all ANN schemes computed over all clusters of $\mathcal{S}^j$ is at most

$$O(m^{1+1/\log p})\frac{f(m)}{m} \leq \tilde{O}(\log\log d)^i \,\mathrm{poly}(d)O(m)^{D+(i+1)/\log p}.$$

Hence, the total space usage is

$$\begin{aligned}&\tilde{O}(\log\log d) \cdot \left(\tilde{O}(\log\log d)^i \cdot \mathrm{poly}(d) \cdot O(m)^{D+(i+1)/\log p}\right) + \mathrm{poly}(d) \cdot \tilde{O}(m)\\ &\leq \tilde{O}(\log\log d)^{i+1} \cdot \mathrm{poly}(d) \cdot O(m)^{D+(i+1)/\log p},\end{aligned}$$

completing the proof. ◁

Finally, we use Claim 4.4 for $m = n$ and $i = \log p$, and obtain that the space usage of the entire ANN data structure is bounded by

$$\tilde{O}(\log\log d)^{\log p} \,\mathrm{poly}(d) \cdot n^{D+\log p/\log p} \leq \mathrm{poly}(d) \cdot n^{D+1} \leq \mathrm{poly}(dn),$$

which completes the proof of Theorem 1.2. ◀

▶ Remark 4.5. Modifying the parameter β of the sparse cover in the proof of Theorem 1.2 from $\log p$ to $\frac{\log p}{\delta}$ for $0 < \delta < 1$ yields a data structure with a slightly larger approximation factor $p^{O(1)+\log(1/\delta)+\log\log p}$, but space requirement that matches that of [18] up to subpolynomial factors in d and an additional n^δ term.

▶ Remark 4.6. The same technique used in the proof of Theorem 1.2, namely applying Theorem 4.2 to construct covers in the preprocessing phase, can also be used to improve the space requirement of ANN for general normed spaces from [8, Theorem 3]. More specifically, for every $0 < \delta < 1$, one can shave an $\Omega(n^{1-\delta})$ factor from the space of their data structure, at the cost of an additional $O(\delta^{-1})$ factor in the approximation.

References

1 A. Andoni and P. Indyk. Nearest neighbors in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry*, chapter 43, pages 1135–1150. CRC Press, 3rd edition, 2017. doi:10.1201/9781315119601.

2 A. Andoni, P. Indyk, and R. Krauthgamer. Overcoming the l_1 non-embeddability barrier: algorithms for product metrics. In *19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 865–874. SIAM, 2009. doi:10.1137/1.9781611973068.94.

3 Alexandr Andoni. *NN search: the old, the new, and the impossible*. PhD thesis, Massachusetts Institute of Technology, 2009. URL: https://hdl.handle.net/1721.1/55090.

4 Alexandr Andoni, Dorian Croitoru, and Mihai Pătraşcu. Hardness of nearest neighbor under l-infinity. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pages 424–433. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.89.

5 Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science*, pages 459–468. IEEE, 2006. doi:10.1109/FOCS.2006.49.

6 Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pages 3287–3318, 2019. doi:10.1142/9789813272880_0182.

7 Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Data-dependent hashing via nonlinear spectral gaps. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 787–800. ACM, 2018. doi:10.1145/3188745.3188846.

8 Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 159–169. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00024.

9 Alexandr Andoni, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Approximate nearest neighbors beyond space partitions. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1171–1190. SIAM, 2021. doi:10.1137/1.9781611976465.72.

10 Alexandr Andoni and Negev Shekel Nosatzki. Embeddings into similarity measures for nearest neighbor search. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 864–893. IEEE, 2025. doi:10.1109/FOCS63196.2025.00045.

11 B. Awerbuch and D. Peleg. Sparse partitions. In *31st Annual IEEE Symposium on Foundations of Computer Science*, pages 503–513, 1990. doi:10.1109/FSCS.1990.89571.

12 Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear time construction of sparse neighborhood covers. *SIAM Journal on Computing*, 28(1):263–277, 1998. doi:10.1137/S0097539794271898.

13 Yiqiao Bao, Anubhav Baweja, Nicolas Menand, Erik Waingarten, Nathan White, and Tian Zhang. Average distortion sketching. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 277–302. IEEE, 2025. doi:10.1109/FOCS63196.2025.00019.

14 Yair Bartal and Lee-Ad Gottlieb. Approximate nearest neighbor search for ℓ_p-spaces ($2 < p < \infty$) via embeddings. *Theoretical Computer Science*, 757:27–35, 2019. doi:10.1016/j.tcs.2018.07.011.

15 Yoav Benyamini and Joram Lindenstrauss. *Geometric nonlinear functional analysis*, volume 48. American Mathematical Soc., 1998.

16 Timothy M. Chan. Approximate nearest neighbor queries revisited. *Discret. Comput. Geom.*, 20(3):359–373, 1998. doi:10.1007/PL00009390.

17 Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *20th annual symposium on Computational geometry*, pages 253–262. ACM, 2004. doi:10.1145/997817.997857.

18 Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory Comput.*, 8(1):321–350, 2012. doi:10.4086/TOC.2012.V008A014.

19 P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *30th Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998. doi:10.1145/276698.276876.

20 Piotr Indyk. On approximate nearest neighbors under ℓ_∞ norm. *J. Comput. Syst. Sci.*, 63(4):627–638, 2001. doi:10.1006/JCSS.2001.1781.

21 Michael Kapralov and Rina Panigrahy. NNS lower bounds via metric expansion for ℓ_∞ and EMD. In *39th International Colloquium on Automata, Languages and Programming, ICALP 2012*, volume 7391 of *Lecture Notes in Computer Science*, pages 545–556. Springer, 2012. doi:10.1007/978-3-642-31594-7_46.

22 Robert Krauthgamer and Nir Petruschka. Lipschitz Decompositions of Finite ℓ_p Metrics. In *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66:1–66:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SoCG.2025.66.

23 Robert Krauthgamer, Nir Petruschka, and Shay Sapir. The power of recursive embeddings for ℓ_p metrics. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 2546–2554. IEEE, 2025. doi:10.1109/FOCS63196.2025.00132.

24 Deepanshu Kush, Aleksandar Nikolov, and Haohua Tang. Near neighbor search via efficient average distortion embeddings. In *37th International Symposium on Computational Geometry, SoCG 2021*, volume 189 of *LIPIcs*, pages 50:1–50:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.SOCG.2021.50.

25 E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000. doi:10.1137/S0097539798347177.

26 Assaf Naor and Kevin Ren. Optimal randomized clustering for subsets of L_p when $p > 2$. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 984–995. SIAM, 2026. doi:10.1137/1.9781611978971.40.

27 Huy L. Nguyen. Approximate nearest neighbor search in ℓ_p. *CoRR*, abs/1306.3601, 2013. arXiv:1306.3601.

28 Daren Yu, Xiao Yu, and Anqi Wu. Making the nearest neighbor meaningful for time series classification. *Proceedings of the 4th International Congress on Image and Signal Processing, CISP 2011*, 5:2481–2485, October 2011. doi:10.1109/CISP.2011.6100672.

Non-Dissective Coverings by Planks

Andrey Kupavskii
Moscow Institute of Physics and Technology, Russia
Saint-Petersburg University, Russia
Innopolis University, Tatarstan Republic, Russia

János Pach
Alfréd Rényi Institute of Mathematics, Hungary

Abstract

A *plank* is the part of space between two parallel planes. The following open problem, posed 45 years ago, can be viewed as the converse of Tarski's plank problem (Bang's theorem): Is it true that if the total width of a collection of planks is sufficiently large, then the planks can be individually translated to cover a unit ball B?

A translative covering of B by planks is said to be *non-dissective* if the planks can be added one by one, in some order, such that the uncovered part remains connected at each step and is empty at the end. Improving a classical result of Groemer, we show that every set of $C/\epsilon^{7/4}$ planks of width ϵ admits a non-dissective translative covering of a 3-dimensional ball B^3, provided C is large enough. Our proof yields a low-complexity algorithm. We also show that $c/\epsilon^{4/3}$ planks are, in general, insufficient for a non-dissective covering of B^3. This provides the first non-trivial lower bound for this problem.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Tarski's plank problem, translative cover, non-dissective cover

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.67

Funding *Andrey Kupavskii*: Supported by the Ministry of Science and Higher Education of the Russian Federation, project No. FSMG-2024-0025.
János Pach: Supported by NKFIH grant K-131529 and ERC Advanced Grant 882971 "GeoScape."

1 Introduction

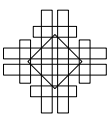
A *plank of width* ϵ in $\mathbb{R}^d$ is a closed region P bounded by two parallel hyperplanes at distance ϵ from each other. The distance between these two hyperplanes is called the *width* of P.

In 1932, Tarski [17] posed the question now known as the *plank problem*: if a convex body K in $\mathbb{R}^d$ is covered by a collection of planks, is it true that the sum of their widths is at least as large as the minimal *width* of K, that is, the minimum width of a plank that contains K? Twenty years later, this question was answered affirmatively by Bang [3, 4]; see also Fenchel [8] for an alternative proof. Bang also conjectured that a similar statement is true with "relative widths" with respect to K. This was proved by Ball [2] for centrally symmetric bodies, but in the general case the problem is open. A spherical analogue of the plank problem, raised by L. Fejes Tóth [7], was settled by Jiang and Polyanskii [12] (see also Ortega-Moreno [16]). A. Bezdek [5] conjectured that the following planar variant of Tarski's conjecture is also true: the total width of a set of planks that cover a unit disk with a small circular hole around its center is at least 2; see also [19, 1]. For many other related open problems, consult [6], Section 3.4. For a recent survey on the topic, see [18].

The following natural question was asked independently by Groemer [9] and Makai-Pach [15]. Can one reverse Bang's theorem? Is it true that if the total width of a collection of planks $P_1, P_2, \ldots$ in $\mathbb{R}^d$ is large enough, then each P_i can be translated into a new position



© Andrey Kupavskii and János Pach;
licensed under Creative Commons License CC-BY 4.0



P_i' such that $P_1', P_2', \ldots$ together cover, say, a ball B^d of *unit radius*? In other words, does there exist a constant C_d such that any collection of planks of total width at least C_d permits a *translative covering* of the unit ball B^d? In the plane, this was proved independently by Makai-Pach [15] and Erdős-Straus (unpublished); see also [10, 11]. For $d \geq 3$, the problem is open. In particular, we have the following.

▶ **Conjecture 1** ([15]). *For every $d \geq 3$, there exists a constant C_d such that every collection of $C_d\epsilon^{-1}$ planks of width ϵ in $\mathbb{R}^d$ permits a translative covering of the unit ball.*

For a long time, the best known result was due to Groemer [9], who designed a greedy algorithm to construct a translative covering of the unit ball for any collection of $O(\epsilon^{-(d+1)/2})$ planks of width ϵ in $\mathbb{R}^d$. Kupavskii and Pach [13, 14] came close to proving Conjecture 1: they showed that the statement is true for any system of only $O(\epsilon^{-1}\log(\epsilon^{-1}))$ planks.

Nevertheless, Groemer's algorithm had a very interesting additional feature. Given a sequence of planks $P_1, \ldots, P_f$, by a simple greedy method, one can find suitable translates P_i' of P_i $(i = 1, \ldots, f)$, with the property that

$$B^d \setminus \cup_{i=1}^{j} P_i'$$

is a *connected* set for every $j \geq 1$. Moreover, Groemer showed that if we insist on this property and we also fix the order of the planks, then the bound $O(\epsilon^{-(d+1)/2})$ is essentially best possible.

▶ **Definition 2.** We say that a collection of planks $P_1, \ldots, P_f$ permits a *non-dissective translative covering* of a convex body K in $\mathbb{R}^d$ if there exist a permutation

$$\sigma : \{1, \ldots, f\} \to \{1, \ldots, f\}$$

and suitable translates P_i' of P_i with the property that

$$K \setminus \cup_{i=1}^{j} P_{\sigma(i)}'$$

is a *connected* set for $j = 1, \ldots, f$, and is empty for $j = f$.

In the present paper, we study the following variant of the problem addressed in Conjecture 1.

▶ **Problem 3.** *What is the smallest integer $f = f(d, \epsilon)$ such that every collection of f planks of width ϵ in $\mathbb{R}^d$ permits a non-dissective translative covering of the unit ball $\mathbb{R}^d$?*

The Makai–Pach covering [15] is non-dissective, and thus we have $f(2, \epsilon) = \Theta(1/\epsilon)$. For every $d \geq 3$, we have

$$c_d\epsilon^{-1} \leq f(d, \epsilon) \leq C_d\epsilon^{-(d+1)/2}. \tag{1}$$

Here, we concentrate on the first open case $d = 3$. Our main theorem improves both the lower and upper bounds in (1).

▶ **Theorem 4.** *There exist constants $c, C > 0$ such that*

$$c\epsilon^{-4/3} \leq f(3, \epsilon) \leq C\epsilon^{-7/4}.$$

In particular, the lower bound shows that Conjecture 1 does not remain true if we restrict our attention to non-dissective coverings.

In the next two sections, we establish the lower and upper bounds in Theorem 4, respectively. We conclude this note with a few remarks (see Section 5).

2 Proof of Theorem 4 – Lower bound

In this section, we establish the lower bound in Theorem 4. Consider $k = \Omega(\epsilon^{-4/3})$ points on the sphere such that the angular distance between any two of them is at least $10\epsilon^{2/3}$, and each point is within angle 1° from the north pole.

Let $P_1, \dots, P_k$ be a sequence of planks of width ϵ, whose normal vectors correspond to the above k points. Consider a non-dissective covering of the unit ball B^3 using planks P_i in the order of increasing i. Since all planks are almost horizontal, we may talk about their "upper" and "lower" boundary planes. Formally, the z-coordinate of the intersection of the upper boundary plane with the z-axis is larger than that for the lower boundary plane.

Let $K_0 = B^3$ and, for all i, $1 \le i \le k$, let $K_i := K_{i-1} \setminus P'_i$, where P'_i is a suitable translate of P_i used for the covering. By the assumption that the covering is non-dissective, K_i is a connected set for every i. As soon as we cover a point of B^3 within angular distance 1° from the equator, we stop the covering process, and we declare that the covering was *successful*. We want to show that the number of planks used up to this moment is $\Omega(\epsilon^{-4/3})$.

We may assume recursively that P'_{i+1}, the translate of P_{i+1} that we use for the covering, satisfies one of the following two conditions:

1. either the upper boundary plane of P'_{i+1} coincides with the upper supporting plane of K_i,
2. or the lower boundary plane of P'_{i+1} coincides with the lower supporting plane of K_i.

Assume without loss of generality that the first plank that covers a point at distance at most 1° from the equator satisfies condition 1 above. Note that this implies that at this moment we have already covered a portion of the upper half of B^3 whose volume is $\Omega(1)$. Slightly abusing notation, we denote the planks that were used for the covering of the upper hemisphere also by $P'_1, \dots, P'_k$.

Note that the normal vectors at the boundary of the part of K_i covered by P'_{i+1} have angle at most 90° with the normal vector defining P'_{i+1}. This holds for all flat faces since all P'_j have normals within angle 1° from the north pole. This also holds for the parts of the boundary coming from B^3 since in the covering procedure we only cover points that are more than 1° from the equator. Using this, we can bound the volume of K_i that is covered by P'_{i+1}: it is at most ϵ times the area of the face of K_{i+1} lying on the boundary of the plank P'_{i+1}. Note that this face is the intersection of the lower boundary plane π^L of P'_{i+1} with K_i. Let X denote this intersection.

▶ **Lemma 5.** *X does not contain a disk of radius $10\epsilon^{1/3}$.*

The lemma implies the lower bound as follows. The diameter of X is at most 2, which is the diameter of B^3. Combining this fact with Lemma 5, we obtain that the area of X is $O(\epsilon^{1/3})$. This, in turn, implies that, for every i, $1 \le i \le k$, the volume of the intersection $P'_{i+1} \cap K_i$ is $O(\epsilon^{4/3})$. The volume of the upper part of B^3 that we need to cover is $\Omega(1)$. Hence, we need $\Omega(\epsilon^{-4/3})$ planks to exhaust this volume. We note that Lemma 5 is one part of the proof that is specific to the non-dissective case.

Proof of Lemma 5. Let n stand for the normal vector to π^L. For any point w in X, we denote by w' the point on the boundary of K_i that is obtained from w by translating it parallel to n inside P'_{i+1}. Note that w' and w are at most ϵ apart.

Let us show that X does not contain a disk D of radius $10\epsilon^{1/3}$. See Figure 1. Assume the contrary, and let v be the center of such a disk D.

Consider the circle C in π^L, concentric to D and of radius $5\epsilon^{1/3}$. Take a point $w \in C$. Consider the normal vector $n_{w'}$ of a supporting plane for K_i at w', whose choice we explain below. If w' lies on the boundary of, potentially, several planks P'_j with $j \le i$, then take a supporting hyperplane of one of these planks P'_j. Otherwise, w' must lie on the boundary of the initial ball B^3, and then we simply take the unique supporting plane to the ball at w'.

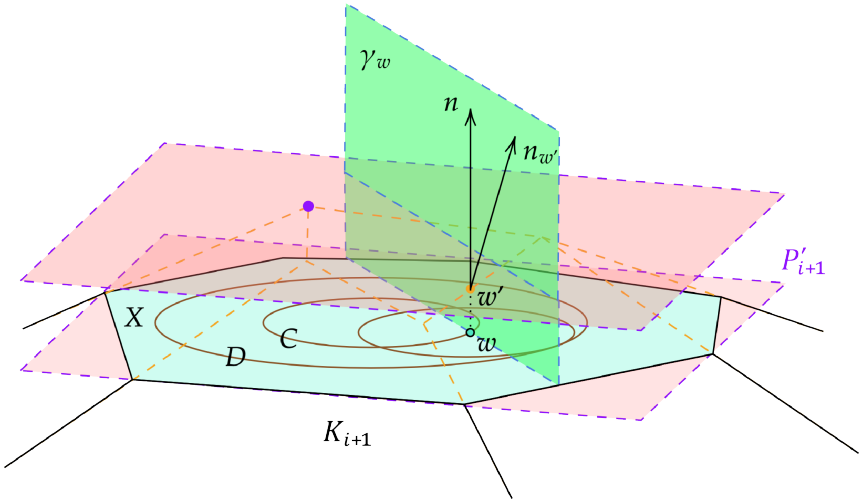


Figure 1 Discs D, C on the face X of the set K_{i+1}. The plank P'_{i+1} is shown by two parallel planes in semi-transparent red. The part of K_i that was cut off by P'_{i+1} is indicated in dashed orange. The plane γ_w containing w, w' and normal vectors $n, n_{w'}$ is semi-transparent green.

Consider the intersection Q of $K_i \cap P'_{i+1}$ with the plane γ_w spanned by n and $n_{w'}$ and passing through w. By our assumption, X contains a disk of radius $5\epsilon^{1/3}$ centered at w. Thus, Q is a convex set of height at most ϵ and with the "base" being a straight line segment orthogonal to n that passes through w and has length at least $5\epsilon^{1/3}$ in each direction from w. Thus, the angle between n and $n_{w'}$ is at most $\epsilon^{2/3}$: otherwise, the supporting line at w' corresponding to $n_{w'}$ will hit the base segment at distance at most $\epsilon \cdot \cot(\epsilon^{2/3}) < 2\epsilon^{1/3}$ from w, which is a contradiction. See Figure 2.

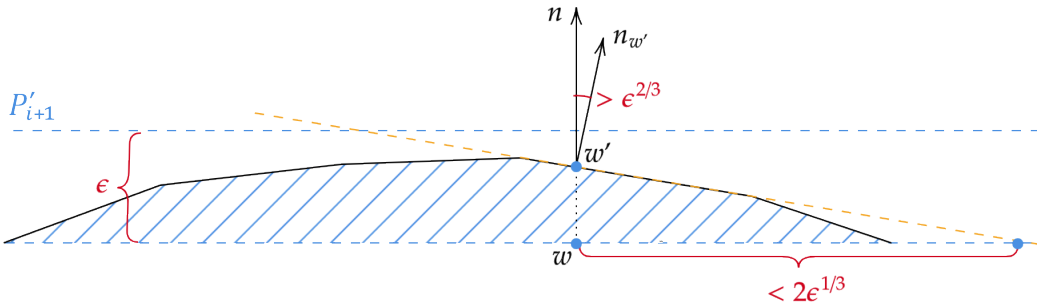


Figure 2 An illustration to the proof of Lemma 5: If the angle between n and $n_{w'}$ is greater than $\epsilon^{2/3}$, then the supporting line at w' hits the "base" $\pi^L \cap \gamma_w$ at distance at most $2\epsilon^{1/3}$.

We conclude that $n_{w'}$ has angle at most $\epsilon^{2/3}$ with n for any $w \in C$. Since the angle between the normal vectors of any two planks is at least $10\epsilon^{2/3}$, $n_{w'}$ cannot be a normal vector to any other plank. Running the same argument for each point $w \in C$, we conclude that the set $C' := \{w' : w \in C\}$ lies on the boundary of B^3. But then the angle between the normal vectors at two points of C' corresponding to the opposite points of C is $\Omega(\mathrm{diam}(C)) = \Omega(\epsilon^{1/3})$, a contradiction with the fact that each of them has angle at most $\epsilon^{2/3}$ with n. ◀

3 Proof of Theorem 4 – Upper bound

Consider k planks of width ϵ, and write k in the form $k = \epsilon^{-\beta}$ for some $\beta > 0$. Let $\alpha > 0$ be a constant to be specified later. By averaging over all directions, we can find a vector n such that there are $\Theta(\epsilon^{2\alpha-\beta})$ planks whose normal vectors form an angle of at most ϵ^α with n. Let these planks be $P_1, \dots, P_m$. (At the end of the proof we will see that for our arguments to work, the optimal choice of the parameters will be $\beta \approx 7/4$ and $\alpha \approx 3/4$.)

Choose a coordinate system in which the direction of the positive z-axis, pointing upwards, is n. Since the normal vectors of the planks $P_1, \dots, P_m$ are very close to n, we can call these planks "almost horizontal", and we can talk about their "upper" and "lower" boundary planes.

Given a convex body K, the *(outer) parallel body* of K at distance 1 is defined as

$$B(K) := \{v \in \mathbb{R}^3 : \inf_{w \in K} \|v - w\| \le 1\}.$$

Clearly, $B(K)$ is a smooth body, hence in every direction it has precisely two supporting planes.

We are going to process the planks one by one. Set $K_0 = B^3$. Let P_1' denote the translate of P_1 whose upper boundary plane is tangent to K_0 from above. Informally, this means that we move P_1 from above as long as we can, preserving the property that $K_0 \setminus P_1'$ remains connected. Since $K_0 = B^3$, we have $B(K_0) = 2B^3$. Denote by P_1'' the translate of P_1 whose upper boundary plane is tangent from above to $B(K_0)$.

Next, we set $K_1 := K_0 \setminus P_1'$. Let P_2' denote the translate of P_2 whose upper boundary plane is tangent to K_1 from above. Analogously, denote by P_2'' the translate of P_2 whose upper boundary plane is tangent to $B(K_1)$ from above. Let $K_2 := K_1 \setminus P_2'$. Proceeding like this, after performing j steps ($j \le m$), we obtain a sequence of convex bodies $K_0 = B^3 \supset K_1 \supset K_2 \supset \dots \supset K_j$, and two sequences of planks, $P_1', \dots, P_j'$ and $P_1'', \dots, P_j''$ with the following properties:

1. $B(K_0) \supset B(K_1) \supset \dots \supset B(K_j)$
2. For every i, $1 \le i \le j$, the distance between the lower (resp. upper) boundary planes of P_i' and P_i'' is 1.
3. If $K_j = K_{j-1} \setminus P_j'$ is *empty*, we terminate the process.

By definition, for every j, we have $K_j = K_0 \setminus \cup_{i=1}^j P_i'$. We will keep track of the (decreasing) sequence of volumes Vol $B(K_i)$. If for some j, we have

$$\text{Vol } B(K_j) < \text{Vol } B^3, \tag{2}$$

then we can conclude that K_j is empty, i.e., the initial ball B^3 is completely covered by the planks $\cup_{i=1}^j P_i'$.

We need some easy observations.

▶ **Proposition 6.** *Let π_i', π_i'' be the lower boundary planes of P_i', P_i'', respectively, and let Π_i be the upper half-space bounded by the π_i''.*

Then, for $i = 1, 2, \dots$, we have

$$B(K_i) = B(K_{i-1} \setminus P_i') \subset B(K_{i-1}) \setminus \Pi_i \subset B(K_{i-1}) \setminus P_i''.$$

Proof. We start by verifying the first inclusion. Let v be any point of $B(K_{i-1} \setminus P_i')$. By definition, there exists $w \in K_{i-1} \setminus P_i'$ with $\|v - w\| \le 1$. Thus, v also belongs to $B(K_{i-1})$. We only have to argue that v does not lie inside the half-space Π_i. If it did, by property 2

above, it would be farther away than 1 from the lower boundary plane of P_i', which separates $K_{i-1} \setminus P_i'$ from v. This would contradict our assumption that $v \in B(K_{i-1} \setminus P_i')$.

The second inclusion holds since $P_i'' \subset \Pi_i$. ◀

Recall that, according to our conventions, the direction of the positive z-axis of the coordinate system coincides with the vector n. At the beginning, we fixed a vertical direction n, but in the sequel we will also use the following simple fact in other scenarios.

▶ **Proposition 7.** *Let K be a convex body and let s be a boundary point of the parallel body $B(K)$ such that the z-coordinate of s is maximum.*

Then $B(K)$ contains a ball U of unit radius *that touches the boundary of $B(K)$ at s.* ◀

Now we quickly outline our proof strategy for the upper bound in Theorem 4. Recall that we selected a "dense" subset of almost parallel directions, in terms of two constants α and β. Our first and main goal is to bound from below the volume of the parallel body $B(K_j)$ of the part of B that we cover with the first j planks corresponding to the chosen directions. To get this lower bound, we will have two cases to consider, and the parameter α is chosen to balance the bound in the two cases. We cover a substantial part of $K_0 = B^3$ using these planks, and remove them. This constitutes the first stage of our covering algorithm.

At the second stage, we repeat the same procedure. This time, in the remaining set of directions we find a "dense" subset (using the same α), and we use the corresponding almost parallel planks to cover the remaining part of the ball. While at stage 1, we started with $K_0 = B^3$, at stage 2 we start where we left off at stage 1. That is, our new initial body $K_0^{(2)}$ will be identical to the last K_j at stage 1. Again, we bound from below the volumes of the parallel bodies of the new shrinking sequence

$$K_0^{(2)} \supset K_1^{(2)} \supset K_2^{(2)} \supset \cdots$$

of uncovered bodies. Because of our choice of parameters, it is guaranteed that after performing this procedure a bounded number of times, at some stage we reach a point where (2) holds. At this point, we can be sure that B^3 is entirely covered.

Apply Proposition 7 with $K_0 = B^3$ to obtain a ball $U \subset B(K_0)$ of unit radius that touches the boundary of $B(K_0)$ at its highest point in the vertical direction n. Let ℓ be the vertical line passing through the center of U (which is the highest point of B^3).

We distinguish two cases.

Case A: There is a set of indices $A \subset \{1, \ldots, m\}$ with $|A| \geq m/2$ such that for any $i \in A$, the intersection of ℓ with $B(K_i)$ is at least $\epsilon/2$ shorter than the intersection of ℓ with $B(K_{i-1})$.

Case B: There is a set of indices $Z \subset \{1, \ldots, m\}$ with $|Z| \geq m/2$ such that for any $i \in Z$, the intersection of ℓ with $B(K_i)$ is less than $\epsilon/2$ shorter than the intersection of ℓ with $B(K_{i-1})$.

In **Case A**, we start our argument as follows. To simplify the notation, assume that $m \in A$. Suppose that the center of U is inside $B(K_m)$ and we have $m\epsilon = o(1)$. Using Proposition 6, we obtain that

$$B(K_m) \subset B(K_{m-1}) \setminus \Pi_m \subset B(K_0) \setminus \Pi_m.$$

Since $U \subset B(K_0)$, we also have that

$$U \cap \Pi_m \subset B(K_0) \cap \Pi_m.$$

By the above inclusions, we can bound from below the difference between the volumes of $B(K_m)$ and $B(K_0)$.

$$\begin{aligned}\text{Vol } B(K_0) - \text{Vol } B(K_m) &\geq \text{Vol } B(K_0) - \text{Vol } \big(B(K_0) \setminus \Pi_m\big)\\ &= \text{Vol } \big(B(K_0) \cap \Pi_m\big)\\ &\geq \text{Vol } \big(U \cap \Pi_m\big) = \Omega(m^2\epsilon^2).\end{aligned}$$

We explain the last equality. The set $U \cap \Pi_m$ is a spherical cap that contains a section of the vertical line ℓ of length at least $m\epsilon/4$. The lower boundary plane π''_m of P''_m is almost horizontal: its normal vector has an angle of at most ϵ^α with ℓ. Thus, $U \cap \Pi_m$ contains a spherical cap and, hence, a cone whose height is roughly the same as the length of the portion of ℓ contained in $U \cap \Pi_m$, which is $\Omega(m\epsilon)$. The radius of the circular base of this cone (and of the cap) is $\Omega(\sqrt{m\epsilon})$ (here is where we use the $m\epsilon = o(1)$ assumption). The volume of this cone is $\Omega(m^2\epsilon^2)$, as claimed.

If the center of U is outside $B(K_m)$ or if $m\epsilon = \Omega(1)$, then we have

$$\text{Vol } B(K_0) - \text{Vol } B(K_m) = \Omega(1).$$

Recall that $m = \Theta(\epsilon^{2\alpha-\beta})$. Thus, the volume that we managed to cover in this case is at least

$$\min\big\{\Omega\big(m^2\epsilon^2\big), \Omega(1)\big\} = \min\big\{\Omega\big(\epsilon^{4\alpha-2\beta+2}\big), \Omega(1)\big\}. \tag{3}$$

In **Case B**, we proceed as follows. In view of Proposition 6, the condition implies that for any $i \in Z$, the intersection of ℓ with $B(K_{i-1}) \setminus \Pi_i$ is less than $\epsilon/2$ shorter than that of $B(K_{i-1})$.

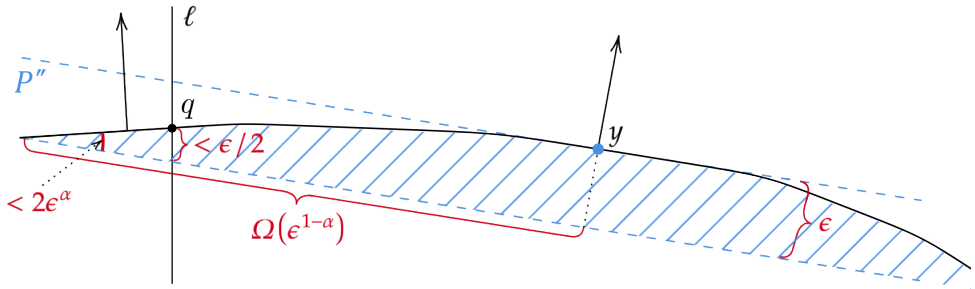


Figure 3 An illustration for the proof of the upper bound in Case B. The convex curve is a portion of the boundary of the intersection of $B(K_{i-1})$ with γ. The shaded area is the part of this intersection covered by the plank P''_i, denoted by Q.

Consider the plane γ spanned by ℓ and a point y at which the upper boundary plane of P''_i touches $B(K_{i-1})$. Set

$$Q := \gamma \cap B(K_{i-1}) \cap P''_i.$$

See Figure 3. If Q intersects ℓ, then let q denote the upper intersection point of the boundary of Q with ℓ. If Q does not intersect ℓ, then let q be the point of Q closest to ℓ, i.e., the endpoint of the "base" of Q closer to ℓ. In both cases, the angle between the normal vector to the boundary of $\gamma \cap B(K_{i-1})$ at q and the normal at y is at most $2\epsilon^\alpha$, because the normal

vectors of all planks $P_1, \dots, P_m$ are within an angle of at most ϵ^α from the vertical direction n (which is the same as the direction of ℓ). The *height* of Q at y, i.e., the intersection of Q with a vertical line through y, is at least ϵ, while its height at q is at most $\epsilon/2$, by the assumption of Case B. This implies that the base of Q has length $\Omega(\epsilon^{1-\alpha})$.

Stepping out of the plane γ, we find that the intersection of P_i'' with $B(K_{i-1})$ cuts out a whole 2-dimensional face. This face contains a disk of radius $\Omega(\epsilon^{1/2})$, since, as before, $B(K_{i-1})$ contains a unit ball that is tangent to the boundary of $B(K_{i-1})$ at the point y. Combining the last two facts, we obtain that the area of the face cut out by P_i'' is $\Omega(\epsilon^{1-\alpha}) \cdot \Omega(\epsilon^{1/2}) = \Omega(\epsilon^{3/2-\alpha})$. Thus, $B(K_{i-1}) \cap P_i''$ contains a cone with base of area $\Omega(\epsilon^{3/2-\alpha})$ and height ϵ. This implies that

$$\text{Vol}\,(B(K_{i-1}) \cap P_i'') = \Omega(\epsilon^{5/2-\alpha}).$$

Therefore, in Case B, we have that

$$\text{Vol}\, B(K_0) - \text{Vol}\, B(K_m) \geq \Omega(m\epsilon^{5/2-\alpha}) = \Omega(\epsilon^{5/2+\alpha-\beta}). \tag{4}$$

To complete the proof of the upper bound in Theorem 4, we iterate the above procedure. Let $k = \epsilon^{-\beta}$. As before, by averaging over directions, for any collection of M planks there exists a direction n such that at least $c\epsilon^{2\alpha}M$ of them have normals within angle ϵ^α of n. In particular, at the first stage we find a subcollection of size $\Theta(\epsilon^{2\alpha-\beta})$ and apply the argument of Cases A and B.

After removing the planks used in this stage, we repeat the same argument on the remaining family. As long as the number of remaining planks is at least $k/2$, the same averaging argument yields at each stage a subcollection of size $\Theta(\epsilon^{2\alpha-\beta})$, so the bounds obtained above apply for each such subcollection.

Since each stage removes at most $O(\epsilon^{2\alpha-\beta})$ planks, after t stages the total number removed is $O(t\epsilon^{2\alpha-\beta})$. Choosing $t = \Omega(\epsilon^{-2\alpha})$ sufficiently small in the constant factor, we ensure that at least $k/2$ planks remain throughout. Thus the procedure can be repeated $\Omega(\epsilon^{-2\alpha})$ times.

Combining (3) and (4), each stage removes volume at least

$$\min\left\{\epsilon^{2+4\alpha-2\beta}, 1, \epsilon^{2\alpha-\beta+5/2-\alpha}\right\}.$$

Therefore the total removed volume of $B(B^3)$ is at least

$$\Omega(\epsilon^{-2\alpha}) \cdot \min\left\{\epsilon^{2+4\alpha-2\beta}, 1, \epsilon^{2\alpha-\beta+5/2-\alpha}\right\} = \Omega\big(\min\left\{\epsilon^{2+2\alpha-2\beta}, \epsilon^{-2\alpha}, \epsilon^{5/2-\beta-\alpha}\right\}\big).$$

Choose α such that $2 + 2\alpha - 2\beta = 5/2 - \alpha - \beta$, that is, $\alpha = \frac{1}{6} + \frac{1}{3}\beta$. Then the above minimum is equal to $\Omega(\min\{\epsilon^{\frac{7}{3}-\frac{4}{3}\beta}, \epsilon^{-\frac{1}{3}-\frac{2}{3}\beta}\})$. In view of (2), if this value is bigger than the volume of $B(B^3)$, then we must have covered the entire ball B^3, and we are done. If we put $\beta = \frac{7}{4}$, then this quantity is $\Omega(1)$. This means that taking $k = C\epsilon^{-7/4}$ with a sufficiently large C is enough to cover B^3.

This completes the proof of the theorem. ◀

4 Concluding remarks

1. In the present note, we focused on the non-dissective covering problem in three dimensions. For $d > 3$, the same lower bound remains valid: indeed, in order to cover a ball in $\mathbb{R}^d$, one needs to cover any of its 3-dimensional sections. One may try to go into the proof and adapt

the lower bound argument to the d-dimensional case. We should then take $\Omega(\epsilon^{-2(d-1)/d})$ points on the sphere with pairwise distances at least $10\epsilon^{2/d}$. The same analysis should lead to a slightly stronger lower bound $\epsilon^{-2(d-1)/d}$. The exponent in the bound does not grow linearly with d, because Lemma 5 guarantees only one "short" direction in the intersection of a plank and the remaining body.

2. The proof of the upper bound can also be extended to $d > 3$, and the resulting bound would be $C_d\epsilon^{-\frac{d+1}{2}+\frac{d-1}{d^2-d+2}}$, which is better than (1), albeit by an additive term in the exponent that tends to 0 as d grows. We believe that neither the lower nor the upper bound is tight.

3. The proof of the upper bound extends to $d > 3$ with only minor changes. Let $k = \epsilon^{-\beta}$ and fix $\alpha > 0$. By averaging over S^{d-1}, there exists a direction n such that at least

$$m = \Theta\left(\epsilon^{(d-1)\alpha-\beta}\right)$$

planks have normals within angle ϵ^α of n, since a spherical cap of angular radius ϵ^α has measure $\Theta(\epsilon^{(d-1)\alpha})$.

We proceed as in Section 3, distinguishing Cases A and B.

Case A. As before, if at least $m/2$ steps reduce the intersection of ℓ with $B(K_i)$ by at least $\epsilon/2$, then we remove from the unit ball a cap of height $\Omega(m\epsilon)$. For $m\epsilon = o(1)$, such a cap has volume $\Omega((m\epsilon)^{(d+1)/2})$. Hence

$$\text{Vol } B(K_0) - \text{Vol } B(K_m) \geq \min\left\{\Omega\left((m\epsilon)^{(d+1)/2}\right), \Omega(1)\right\} = \min\left\{\Omega\left(\epsilon^{\frac{d+1}{2}(1+(d-1)\alpha-\beta)}\right), \Omega(1)\right\}.$$

Case B. The planar argument in γ is unchanged and yields a segment of length $\Omega(\epsilon^{1-\alpha})$. Passing to d dimensions, the boundary of $B(K_{i-1})$ contains a $(d-2)$-dimensional ball of radius $\Omega(\epsilon^{1/2})$, hence the exposed $(d-1)$-dimensional face has measure $\Omega(\epsilon^{d/2-\alpha})$. Therefore

$$\text{Vol } (B(K_{i-1}) \cap P_i'') = \Omega\left(\epsilon^{d/2+1-\alpha}\right),$$

and summing over the m steps gives

$$\text{Vol } B(K_0) - \text{Vol } B(K_m) \geq \Omega\left(m\epsilon^{d/2+1-\alpha}\right) = \Omega\left(\epsilon^{d/2+1+(d-2)\alpha-\beta}\right).$$

Repeating the argument $\Omega(\epsilon^{-(d-1)\alpha})$ times, we obtain a total contribution of order

$$\min\left\{\epsilon^{\frac{d+1}{2}-\frac{d+1}{2}\beta+\frac{(d-1)^2}{2}\alpha}, \epsilon^{-(d-1)\alpha}, \epsilon^{\frac{d}{2}+1-\beta-\alpha}\right\}.$$

Balancing the first and third exponents gives

$$\alpha = \frac{1+(d-1)\beta}{d^2-2d+3}.$$

Substituting and setting the exponent to zero yields

$$\beta = \frac{d+1}{2} - \frac{d-1}{d^2-d+2}.$$

Thus

$$f(d,\epsilon) \leq C_d\,\epsilon^{-\frac{d+1}{2}+\frac{d-1}{d^2-d+2}}.$$

This improves (1), although only by an additive term in the exponent that tends to 0 as $d \to \infty$. We believe that neither the lower nor the upper bound is tight.

4. Our upper bound argument in Section 3 can be easily turned into a polynomial-time algorithm for constructing a cover. Take $k = C\epsilon^{-7/4}$ planks of width ϵ. We first iteratively construct an ordering of the planks which will turn out to guarantee that we end up with a cover. For this, first we (approximately) compute the angle between any two normal vectors and construct a graph G whose vertices are the selected k planks. Two vertices are joined by an edge if the angle between the normal vectors of the corresponding planks is smaller than $\epsilon^\alpha = \epsilon^{3/4}$. Then, at each step, we

(a) find a vertex v whose degree is at least the average degree in G;

(b) include v and its neighbors in the ordering (the order within this set is arbitrary);

(c) modify G by deleting v and its neighbors from G.

In the proof of the upper bound, at each step, we actually work with a chunk of vectors that were included in the ordering in one step.

Finally, we construct the covering, i.e., we find suitable translates of the planks that cover the ball, as follows. For the i-th plank in the ordering, we need to find the maximum (or an approximate maximum) of a linear function over the convex set K that remains from the original ball after removing all points covered by the first $i-1$ planks. Then we modify the set K by including an extra linear inequality in its definition, and repeat the whole procedure, if necessary.

References

1 Gábor Ambrus, Julian Huddell, Maggie Lai, Matthew Quirk, and Elias Williams. Covering spiky annuli by planks. *Discrete & Computational Geometry*, 2025. doi:10.1007/s00454-025-00799-2.

2 Keith Ball. The plank problem for symmetric bodies. *Inventiones Mathematicae*, 104(3):535–543, 1991. doi:10.1007/BF01245089.

3 Thøger Bang. On covering by parallel strips. *Matematisk Tidsskrift B*, pages 49–53, 1950.

4 Thøger Bang. A solution of the plank problem. *Proceedings of the American Mathematical Society*, 2(6):990–993, 1951. doi:10.1090/S0002-9939-1951-0046672-4.

5 András Bezdek. Covering an annulus by strips. *Discrete & Computational Geometry*, 30(2):177–180, 2003. doi:10.1007/s00454-003-0002-y.

6 Peter Brass, William O. J. Moser, and János Pach. *Research Problems in Discrete Geometry*. Springer, New York, 2005. doi:10.1007/0-387-29929-7.

7 László Fejes Tóth. Research problems: Exploring a planet. *American Mathematical Monthly*, 80(9):1043–1044, 1973.

8 Werner Fenchel. On Th. Bang's solution of the plank problem. *Matematisk Tidsskrift B*, pages 49–51, 1951.

9 Helmut Groemer. On coverings of convex sets by translates of slabs. *Proceedings of the American Mathematical Society*, 82(2):261–266, 1981. doi:10.1090/S0002-9939-1981-0609663-7.

10 Helmut Groemer. Covering and packing properties of bounded sequences of convex sets. *Mathematika*, 29(1):18–31, 1982. doi:10.1112/S0025579300012122.

11 Helmut Groemer. Some remarks on translative coverings of convex domains by strips. *Canadian Mathematical Bulletin*, 27(2):233–237, 1984. doi:10.4153/CMB-1984-033-9.

12 Zilin Jiang and Alexandr Polyanskii. Proof of László Fejes Tóth's zone conjecture. *Geometric and Functional Analysis*, 27(6):1367–1377, 2017. doi:10.1007/s00039-017-0427-6.

13 Andrey Kupavskii and János Pach. Simultaneous approximation of polynomials. In *Discrete and Computational Geometry and Graphs (JCDCGG 2015)*, volume 9943 of *Lecture Notes in Computer Science*, pages 193–203, Cham, 2016. Springer. doi:10.1007/978-3-319-48532-4_17.

14 Andrey Kupavskii and János Pach. From Tarski's plank problem to simultaneous approximation. *American Mathematical Monthly*, 124(6):494–505, 2017. doi:10.4169/amer.math.monthly.124.6.494.

15 Endre Makai, Jr. and János Pach. Controlling function classes and covering Euclidean space. *Studia Scientiarum Mathematicarum Hungarica*, 18(2–4):435–459, 1983.

16 Oscar Ortega-Moreno. An optimal plank theorem. *Proceedings of the American Mathematical Society*, 149(3):1225–1237, 2021. doi:10.1090/proc/15228.

17 Alfred Tarski. Uwagi o stopniu równoważności wielokątów. *Parametr*, 2:310–314, 1932.

18 William Verreault. Plank theorems and their applications: A survey. *Bulletin of the London Mathematical Society*, 2025. doi:10.1112/blms.70230.

19 Stuart White and Laura Wisewell. Covering polygonal annuli by strips. *Discrete & Computational Geometry*, 37(4):577–585, 2007. doi:10.1007/s00454-007-1305-z.

Optimal Bounds for Spanners and Tree Covers in Doubling Metrics

An La
University of Massachusetts Amherst, MA, USA

Hung Le
University of Massachusetts Amherst, MA, USA

Shay Solomon
Tel Aviv University, Israel

Cuong Than
University of Massachusetts Amherst, MA, USA

Vinayak
University of Massachusetts Amherst, MA, USA

Shuang Yang
University of Massachusetts Amherst, MA, USA

Tianyi Zhang
State Key Laboratory for Novel Software Technology, Nanjing University, China

Abstract

It is known that any n-point set in the d-dimensional Euclidean space $\mathbb{R}^d$, for $d = O(1)$, admits:

1. A $(1+\varepsilon)$-spanner with maximum degree $\tilde{O}(\varepsilon^{-d+1})$ and with lightness $\tilde{O}(\varepsilon^{-d})$, for any $\varepsilon > 0$.[1]
2. A $(1+\varepsilon)$-tree cover with $\tilde{O}(n \cdot \varepsilon^{-d+1})$ trees and maximum degree of $O(1)$ in each tree.

Moreover, all the parameters in these constructions are optimal: For any $2 \leq d = O(1)$, there exists an n-point set in $\mathbb{R}^d$, for which any $(1+\varepsilon)$-spanner has $\tilde{\Omega}(n \cdot \varepsilon^{-d+1})$ edges and lightness $\tilde{\Omega}(\varepsilon^{-d})$.

The upper bounds for Euclidean spanners rely heavily on the spatial property of *cone partitioning* in $\mathbb{R}^d$, which does not seem to extend to the wider family of *doubling metrics*, i.e., metric spaces of constant *doubling dimension*. In doubling metrics, a **simple spanner construction from two decades ago, the *net-tree spanner***, has $\tilde{O}(n \cdot \varepsilon^{-d})$ edges, and it could be transformed into a spanner of maximum degree $\tilde{O}(\varepsilon^{-d})$ and lightness $\tilde{O}(n \cdot \varepsilon^{-(d+1)})$ by pruning redundant edges. Moreover, a careful refinement of the net-tree spanner yields a $(1+\varepsilon)$-tree cover with $\tilde{O}(\varepsilon^{-d})$ trees.

Despite a large body of work, the problem of obtaining tight bounds for spanners and tree covers in the wider family of *doubling metrics* has remained elusive. We resolve this problem by presenting:

1. A surprisingly simple and tight lower bound, which shows that the net-tree spanner and its pruned version are optimal with respect to all the involved parameters.
2. A new construction of $(1+\varepsilon)$-tree covers with $\tilde{O}(n \cdot \varepsilon^{-d})$ trees, with maximum degree $O(1)$ in each tree. This construction is optimal with respect to the number of trees and maximum degree.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners; Theory of computation → Routing and network design problems; Mathematics of computing → Graph algorithms

Keywords and phrases doubling metrics, doubling spanners, Euclidean spanners, tree cover

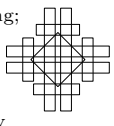
Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.68

Related Version *Full Version*: https://arxiv.org/abs/2508.11555

[1] The *lightness* is a normalized notion of weight, where we divide the spanner weight by the weight of a minimum spanning tree. Here and throughout, the $\tilde{O}$ and $\tilde{\Omega}$ notations hide $\texttt{polylog}(\varepsilon^{-1})$ terms.



© An La, Hung Le, Shay Solomon, Cuong Than, Vinayak, Shuang Yang, and Tianyi Zhang;
licensed under Creative Commons License CC-BY 4.0





Funding *An La*: Supported by the NSF CAREER award CCF-2237288 and an NSF grant CCF-2517033.
Hung Le: Supported by the NSF CAREER award CCF-2237288 and an NSF grant CCF-2517033.
Shay Solomon: Supported by the European Union (ERC, DynOpt, 101043159). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Shay Solomon is also funded by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, and the United States National Science Foundation (NSF).
Cuong Than: Supported by the NSF CAREER award CCF-2237288 and NSF grant CCF-2121952, and a Google Ph.D. Fellowship.
Vinayak: Supported by the NSF CAREER award CCF-2237288 and an NSF grant CCF-2517033.
Shuang Yang: Supported by the NSF CAREER award CCF-2237288 and an NSF grant CCF-2517033.
Tianyi Zhang: Supported by Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118) and the "111 Center" (No. B26023).

1 Introduction

1.1 Low-Dimensional Euclidean Spaces

1.1.1 Euclidean Spanners

Let P be a set of n points in the Euclidean space $\mathbb{R}^d, d \geq 2$, and consider the complete weighted graph $G_P = (P, \binom{P}{2}, \|\cdot\|)$ induced by P, where the weight of any edge $(x, y) \in \binom{P}{2}$ is the Euclidean distance $\|xy\|$ between its endpoints. We say that a spanning subgraph $H = (P, E, \|\cdot\|)$ of G_P (with $E \subseteq \binom{P}{2}$) is a *t-spanner* for P, for a parameter $t \geq 1$ that is called the *stretch* of the spanner, if $d_H(x, y) \leq t \cdot \|xy\|$ holds $\forall x, y \in P$. Spanners for Euclidean spaces, or *Euclidean spanners*, were introduced in the pioneering work of Chew [14] from 1986, which gave an $O(1)$-spanner with $O(n)$ edges. The first constructions of Euclidean $(1+\varepsilon)$-spanners, for any parameter $\varepsilon > 0$, were given in the seminal works of [15, 34, 35]) that introduced the *Θ-graph* in 2 and 3-dimensional Euclidean spaces, which was generalized for any Euclidean space $\mathbb{R}^d$ in [45, 2]. The Θ-graph is a natural variant of the *Yao graph*, introduced by Yao [49] in 1982, and can be described as follows.

> **Yao graph:** $\forall p \in P$, the space $\mathbb{R}^d$ around p is partitioned into *cones* of angle Θ each ($O(\Theta^{-d+1})$ cones for $d = O(1)$), and then edges are added between point p and its closest point in each of these cones.

The Θ-graph is defined similarly to the Yao graph: instead of connecting p to its closest point in each cone, connect it to a point whose *orthogonal projection* to some fixed ray contained in the cone is closest to p. Taking Θ to be $c\varepsilon$, for small enough constant c, one can show that the stretch of the Θ and Yao graphs is at most $1+\varepsilon$. Since the number of cones is asymptotically ε^{-d+1} (for $d = O(1)$), the *maximum degree* is $O(\varepsilon^{-d+1})$, leading to a total of $O(n \cdot \varepsilon^{-d+1})$ edges.

The tradeoff between stretch $1+\varepsilon$ and $O(n \cdot \varepsilon^{-d+1})$ edges (and maximum degree $O(\varepsilon^{-d+1})$) is also achieved by other constructions, including the *greedy spanner* [2, 12, 43] and the gap-greedy spanner [46, 4]. The spatial cone partitioning of $\mathbb{R}^d$ is key to attaining the size bound $O(n \cdot \varepsilon^{-d+1})$ in these constructions, either in the constructions themselves or in their analysis. In 2019, Le and Solomon [38] showed that this stretch-size tradeoff is *existentially*

tight: For any constant $d \geq 2$, there exists an n-point set in $\mathbb{R}^d$ (basically a set of evenly spaced points on the d-dimensional sphere), for which any $(1+\varepsilon)$-spanner has $\Omega(n \cdot \varepsilon^{-d+1})$ edges.

Light spanners. Another basic property of spanners is *lightness*, defined as the ratio of a spanner's total weight to the weight of the Minimum Spanning Tree of P. A long line of work [2, 16, 18, 44, 43, 6, 38], starting from the paper of Das et al. [16] in 1993, showed that for any point set in $\mathbb{R}^d$, the greedy $(1+\varepsilon)$-spanner of [2] has constant (depending on ε and d) lightness. The exact dependencies on ε and d in the lightness bound were not explicated in [2, 16, 18]. In their seminal work on approximating TSP in $\mathbb{R}^d$ using light spanners, Rao and Smith [44] showed that the greedy spanner has lightness $\varepsilon^{-O(d)}$, and they raised the question of determining the exact constant hiding in the exponent of their upper bound. The proofs in [2, 16, 18, 44] were incomplete; the first complete proof was given in [43], where a lightness bound of $O(\varepsilon^{-2d})$ was established. This line of work culminated with the work of Le and Solomon [38], which improved the lightness bound to $\tilde{O}(\varepsilon^{-d})$, where we shall use the $\tilde{O}$ and $\tilde{\Omega}$ notations to suppress $\texttt{polylog}(\varepsilon^{-1})$ terms. The exact lightness bound here is $O(\varepsilon^{-d} \cdot \log(1/\varepsilon))$, but for the sake of brevity we will mostly disregard $\texttt{polylog}(\varepsilon^{-1})$ terms from now on. They [38] also showed that this stretch-lightness tradeoff is *existentially tight* (up to a $\log(\varepsilon^{-1})$ factor): for any constant $d \geq 2$, there exists an n-point set in $\mathbb{R}^d$ (the same set of evenly spaced points on the d-dimensional sphere), for which any $(1+\varepsilon)$-spanner has lightness $\Omega(\varepsilon^{-d})$.

1.1.2 Euclidean Tree Covers

A more structured variant of spanners is the *tree cover*, defined as a collection $\mathcal{T}$ of trees for a given graph or metric space such that, for every pair of vertices, there exists a tree in $\mathcal{T}$ that preserves their distance up to a given stretch factor, without shortening distances. Due to their strong structural properties, tree covers play a pivotal role in routing, (path-reporting) distance oracles, and various other algorithmic applications. The first tree cover construction for a graph or metric family was given by Arya et al. [3] – known as the "Dumbbell Theorem" – asserting that any low-dimensional Euclidean space admits a $(1+\varepsilon)$-stretch tree cover of size $O(\varepsilon^{-d} \cdot \log(1/\varepsilon))$, for any $\varepsilon > 0$. The lower bound by Le and Solomon [37] on the size of spanners directly implies that any $(1+\varepsilon)$-stretch tree cover must have size at least $\Omega(\varepsilon^{-d+1})$. Recently, Chang et al. [13] closed this longstanding gap between the upper and lower bounds on the size of tree covers, up to a $\log(1/\varepsilon)$ factor, by improving the size upper bound to $O(\varepsilon^{-d+1} \cdot \log(1/\varepsilon))$. Another important quality measure of tree covers is the maximum degree of a vertex in any tree; in particular, a constant degree tree cover enables compact routing schemes with small routing tables [13]. Chang et al. constructed a constant degree tree cover without increasing the size beyond $O(\varepsilon^{-d+1} \cdot \log(1/\varepsilon))$. Whether the $\log(1/\varepsilon)$ factor separating this upper bound from the lower bound is necessary remains an open problem.

1.2 Doubling Metrics

1.2.1 Doubling Spanners

Euclidean spanners have been extensively studied over the years [34, 35, 2, 17, 3, 18, 4, 44, 26, 22, 1, 7, 24, 9, 19, 48, 20, 38], with a plethora of applications, such as in geometric approximation algorithms [44, 27, 30, 28], geometric distance oracles [27, 30, 29, 28], network design [32, 42] and machine learning [25]. (See the book [43] for an excellent account on

Euclidean spanners and their applications.) There is a growing body of work on *doubling spanners*, i.e., spanners for the wider family of *doubling metrics*;[2] see [22, 8, 7, 31, 23, 24, 47, 10, 20, 11, 48, 6, 41, 33, 40], and the references therein. **A common theme in this line of work is to devise constructions of spanners for doubling metrics that are just as good as the analog Euclidean spanner constructions**. Alas, this may not always be possible, as doubling metrics do not possess the spatial properties of Euclidean spaces – and in particular the spatial property of *cone partitioning*, which is key to achieving the aforementioned stretch-size and stretch-lightness upper bounds. Despite this shortcoming, the basic *packing bound* in doubling metrics can be used to construct, via a simple greedy procedure, a *hierarchy of nets*, which induces the so-called *net-tree* [31, 22, 8]. (See Section 2 for the packing bound and other definitions.) Equipped with such a hierarchy of nets, a $(1+\varepsilon)$-spanner with $\tilde{O}(n \cdot \varepsilon^{-d})$ edges is constructed as follows [31, 22, 8].

Net-tree spanner [31, 22, 8]: Let $X = N_0 \supseteq N_1 \supseteq N_2 \supseteq \ldots \supseteq N_{\lceil \log \Delta \rceil}$ be a hierarchy of nets of a doubling metric (X, d_X) with minimum distance 1 and maximum distance Δ, where N_i is a 2^i-net for N_{i-1}, $1 \leq i \leq \lceil \log \Delta \rceil$.
Let $E_{-1} = \emptyset$ and for each $0 \leq i \leq \lceil \log \Delta \rceil$:

$$E_i = \left\{ (x,y) \mid x,y \in N_i, d_X(x,y) \leq \left(4 + \frac{32}{\varepsilon}\right) 2^i \right\} \setminus E_{i-1} \tag{1}$$

Then $(X, \cup_{0 \leq i \leq \lceil \log \Delta \rceil} E_i, d_X)$ is a $(1+\varepsilon)$-spanner of (X, d_X).

The net-tree spanner is a simple and basic spanner construction from over 20 years ago, and it provides the state-of-the-art size bound, $O(n \cdot \varepsilon^{-d} \cdot \log(1/\varepsilon))$. Le and Solomon [39] presented a unified framework for transforming sparse spanners into light spanners by carefully pruning redundant edges. In particular, for doubling metrics, the framework of [39] provides a pruned spanner, with a lightness bound of $\tilde{O}(\varepsilon^{-(d+1)})$, which is the state-of-the-art lightness bound for doubling metrics. Interestingly, these upper bounds on the size and lightness in doubling metrics exceed the respective Euclidean bounds by a factor of ε^{-1}. Despite a large body of work in the area, it has remained a longstanding open question whether the superior Euclidean bounds of $\tilde{O}(n \cdot \varepsilon^{-d+1})$ edges and lightness $\tilde{O}(\varepsilon^{-d})$ could be achieved also in doubling metrics.

▶ **Question 1.** *Can one get a construction of $(1+\varepsilon)$-spanners in doubling metrics with $\tilde{O}(n \cdot \varepsilon^{-d+1})$ edges and/or lightness $\tilde{O}(\varepsilon^{-d})$?*

This question is related to a possibly deeper question, regarding the (im)possibility to generalize the spatial cone partitioning in $\mathbb{R}^d$ to arbitrary doubling metrics.

1.2.2 Doubling Tree Covers

Bartal et al. [5] constructed a tree cover of size $O(\varepsilon^{-d} \cdot \log(1/\varepsilon))$ using the net-tree by redistributing cross edges at each level into color classes. However, in their construction, a vertex in one of the trees may have many children, resulting in unbounded degree, potentially as large as $\log(\Delta)$, where Δ is the aspect ratio. Motivated by the Euclidean setting, it is natural to ask whether one can obtain a tree cover with bounded degree in doubling metrics.

[2] The *doubling dimension* of a metric space is the smallest value d such that every ball B in the metric can be covered by at most 2^d balls of half the radius of B; a metric space is called *doubling* if its doubling dimension is constant. The doubling dimension generalizes the standard Euclidean dimension, as the doubling dimension of the Euclidean space $\mathbb{R}^d$ is $\Theta(d)$.

▶ **Question 2.** *Can one construct a $(1+\varepsilon)$-tree cover in doubling metrics whose maximum degree is an absolute constant, i.e., $O(1)$, while having size $\tilde{O}(\varepsilon^{-d})$?*

1.3 Our contribution

We give a negative answer to Question 1 by a **surprisingly simple** and tight lower bound.

▶ **Theorem 1.** *For any integer constant $d \geq 1$, parameter $\varepsilon \in (0,1)$, and $n \in \mathbb{Z}^+$ satisfying $\varepsilon^{-d} = O(n)$, there exists an n-point ultrametric space (X, d_X) of doubling dimension d such that any $(1+\varepsilon)$-spanner $G = (X, E, w)$ for X has $\Omega(n \cdot \varepsilon^{-d})$ edges and lightness $\tilde{\Omega}(\varepsilon^{-(d+1)})$.*

Since the net-tree spanner has sparsity [3] $O(\varepsilon^{-d})$ [22], our lower bound shows, in particular, that **the net-tree spanner is optimal**; this is our **key conceptual contribution**.

Second, **as our main technical contribution**, we show that a tree cover with constant degree and nearly optimal size is achievable, resolving Question 2 affirmatively.

▶ **Theorem 2.** *For any metric (X, d_X) with doubling dimension d, parameter $0 < \varepsilon < 1$, there exists a $(1+\varepsilon)$-stretch tree cover $\mathcal{T}$ of size $\tilde{O}\left(\varepsilon^{-d}\right)$, where each tree in $\mathcal{T}$ has maximum vertex degree $O(1)$.*

To complement our result, Theorem 3 shows that it is impossible to eliminate the $\mathsf{polylog}(1/\varepsilon)$ factor from the size of any constant-degree tree cover (the detailed proof is provided in Appendix A of the full version of this paper).

▶ **Theorem 3.** *For any integer constant $d \geq 1$, any parameter $\varepsilon \in (0,1)$, and any $n \in \mathbb{Z}^+$ satisfying $\varepsilon^{-d} = O(n)$, there exists an n-point ultrametric space (X, d_X) of doubling dimension d such that any $(1+\varepsilon)$-spanner $G = (X, E, w)$ for X has maximum degree $\Omega(\log(1/\varepsilon) \cdot \varepsilon^{-d})$.*

Theorem 3 demonstrates an inherent multiplicative gap of $\log(1/\varepsilon)$ between the maximum degree and the optimal sparsity of $(1+\varepsilon)$-spanners in doubling metrics, a gap that does not appear in Euclidean spaces. Although Theorem 3 is not the main focus of our work, it provides an interesting separation between Euclidean and doubling metrics. Moreover, it shows that the $\mathsf{polylog}(1/\varepsilon)$ term in Theorem 2 (hidden via the $\tilde{O}$-notation) is unavoidable in doubling metrics, whereas its necessity in Euclidean spaces remains an open question.

1.4 Technical Overview

Our tree cover construction provides the key technical contribution of this work. In what follows, we provide a technical overview of this construction.

Our tree cover construction relies on a hierarchical structure – specifically, the standard *net-tree* – and builds a tree cover so that every cross edge e (i.e., every edge in E_i) is contained in some tree. For every pair of points (u, v), there exists a cross edge (u', v') such that the path from u to v that goes through (u', v') provides a good approximation of $d_X(u, v)$. The distance between u and v is then preserved by the tree T that contains (u', v'); in this case, we say that (u, v) is covered by T.

The construction proceeds in two phases. In Phase 1, we partition the cross edges into congruent classes according to their levels modulo $\log(1/\varepsilon)$, and handle each class independently. For each class, we distribute the cross edges into each tree in our tree cover. Note that after Phase 1, we already obtain a tree cover of stretch $(1+\varepsilon)$ and size $\tilde{O}(\varepsilon^{-d})$.

[3] The *sparsity* of a spanner H is the ratio between number of edges in H over $n-1$, for $n > 1$.

In Phase 2, for a fixed level, we replace edges in a controlled manner to obtain the required bounded degree property, without significantly increasing the stretch. The desired $(1+\varepsilon)$ stretch bound can then be recovered by a simple scaling argument.

Our Phase 2 is inspired by the degree-reduction procedure of [13]. To achieve constant degree, we proceed in two steps. First, using the technique in [8], we reroute the edges in each tree T in the cover produced by Phase 1, obtaining trees of maximum degree $\varepsilon^{-O(d)}$, while the stretch increases only to $1+O(\varepsilon)$. Second, we use the rerouting technique in [13] to obtain absolute constant maximum degree. For each vertex u in each such tree T, we replace the edges connecting u to its children by a binary tree. Suppose that l is the maximum distance from (the representative of) u to its children, this step incurs an additive distortion of $\Theta(\log(\varepsilon^{-d})\cdot l)$ for every pair of points covered by T.

A key technical challenge in integrating Phase 2 into Phase 1 is that the edges replaced by binary trees must be sufficiently short relative to the cross edges at the same level. Otherwise, the additive distortion for a pair (u,v) might become comparable to $d(u,v)$ itself, producing a stretch larger than 2. In the construction of [5], a point u in a tree T may be incident to ε^{-d} cross edges of nearly equal length; applying Phase 2 on top of [5] would therefore not yield a tree cover with good stretch.

Our first idea is to ensure that, in Phase 1, for every level we partition the cross edges into $O(\varepsilon^{-d})$ sets such that in each tree of the resulting cover, every point u is incident to *at most one* edge from each set. This motivates partitioning E_i into sets of *matchings*.

We implement Phase 1 as follows. We initialize tree cover $\mathcal{T}$ with $\tilde{O}(\varepsilon^{-d})$ empty forests. For each congruent class modulo $\log(1/\varepsilon)$, we process the levels in increasing order. For each level i, we partition E_i into sets of matchings $\mathcal{M}_i$. For each matching $M\in\mathcal{M}_i$, we assign M to forest $F\in\mathcal{T}$, and add all edges in M to F. We continue this process until the top level.

A second challenge is ensuring that the graph F remains a forest after inserting the edges of M. Specifically, we guarantee that for any two edges (u,v) and (u',v') in M, there is no tree in the forest F containing both u and u'. This leads to our second idea: each matching M must be *pairwise far*. That is, for every $M\in\mathcal{M}_i$ and every two distinct edges $e,e'\in M$, the distance between (any two endpoints of) e and e' must be at least $\Theta(2^i)$. Under this condition, an inductive argument shows that every tree in each forest F has small diameter $(<2^i)$; thus, after connecting each tree to an edge in M, a sufficient separation still remains, ensuring that the resulting graph is indeed a forest. Such matchings, also known as *red–blue matchings*, were also used by [21, 36] in different contexts.

2 Preliminaries

Let (X,d_X) be a metric space. The *aspect ratio* Δ of X is the ratio of its largest to smallest pairwise distances. The following definitions are crucial for our construction and proofs.

▶ **Definition 4** (r-net). *For a metric space (X,d_X), an r-net $N\subseteq X$ is a set satisfying:*
1. ***[Packing]*** *for every distinct $u,v\in N$, $d_X(u,v)>r$*
2. ***[Covering]*** *for every $x\in X$, there exists $u\in N$ such that $d_X(u,x)\le r$.*

▶ **Definition 5** ((α,β)-Net Tree). *An (α,β)-net tree τ is induced by a hierarchy of nets $N_{-1}=X,N_0,N_1,\ldots,N_{\log\Delta}$ at levels $-1,0,1,\ldots,\log\Delta$ of τ resp., satisfying the following:*
1. *(Hierarchy) $N_i\subseteq N_{i-1}$ for every $i\in[\log\Delta]$.*
2. *(Net) For every $i\in\{0,1,\ldots,\log\Delta\}$, N_i is an $(\alpha\cdot\beta^i)$-net of N_{i-1}.*

The following lemma gives the basic packing bound of doubling metrics.

▶ **Lemma 6** (Packing Lemma). *Metric (X, d_X) has doubling dimension d. If $S \subseteq X$ is a subset with pairwise distance at least r that is contained in a ball of radius R, then $|S| \leq \left(\frac{4R}{r}\right)^d$.*

▶ **Definition 7** (Ultrametric). *An* ultrametric *(X, d_X) is a metric space that satisfies the strong triangle inequality, i.e., for any $u, v, w \in X$, we have $d_X(u, w) \leq \max\{d_X(u, v), d_X(v, w)\}$.*

▶ **Definition 8** (k-HST). *Let $k > 1$ be a fixed constant. A metric space (X, d_X) is a k-*hierarchical well-separated tree (k-HST) *if there exists a bijection ϕ from X to leaves of a rooted tree T in which:*

1. *Each node $v \in T$ is associated with a label $\Gamma(v)$ such that $\Gamma(v) = 0$ if v is a leaf, and $\Gamma(v) \geq k \cdot \Gamma(u)$ $(\Gamma(v) > 0)$ if v is an internal node and u is a child of v.*
2. *$d_X(x, y) = \Gamma(\texttt{LCA}(\phi(x), \phi(y)))$, where $\texttt{LCA}(u, v)$ is the lowest common ancestor of u, v.*

Lastly, we defer some proofs of lemmas/claims (marked by $(\star)$) to the appendix of the full version.

3 Sparsity and Lightness Lower Bounds for $(1 + \varepsilon)$-Spanners.

This section is devoted to the proof of Theorem 1. We state it again below:

▶ **Theorem 1.** *For any integer constant $d \geq 1$, parameter $\varepsilon \in (0, 1)$, and $n \in \mathbb{Z}^+$ satisfying $\varepsilon^{-d} = O(n)$, there exists an n-point ultrametric space (X, d_X) of doubling dimension d such that any $(1 + \varepsilon)$-spanner $G = (X, E, w)$ for X has $\Omega(n \cdot \varepsilon^{-d})$ edges and lightness $\tilde{\Omega}(\varepsilon^{-(d+1)})$.*

We begin by constructing an ultrametric space (X, d_X) for which we later prove that any $(1 + \varepsilon)$-spanner must satisfy the size and lightness lower bound.

Without loss of generality, assume that n is a power of 2^d. More specifically, if n is not a power of 2^d, let $\tilde{n} = (2^d)^\ell$ be the largest integer power of 2^d before n, with $\tilde{n} \geq n/2^d$, and apply the argument to $\tilde{n}$ instead of n. The size lower bound will decrease by at most an $O(1)$ factor 2^d and the lightness lower bound will remain the same.

We define the following 2-HST (X, d_X). Let T be a rooted perfect 2^d-ary tree with n leaf nodes: all leaf nodes are at the same depth $\log_{2^d} n = \frac{\log_2 n}{d}$ and each internal node has 2^d children. The *height* of a node is its distance to the closest leaf, so the leaf nodes have height 0, their parents have height 1, and so on, until reaching the root, at height $\frac{\log_2 n}{d}$. For each node v in the tree we assign a label, denoted by $\Gamma(v)$. Each leaf node is assigned label 0, and each node of height $i = 1, \ldots, \frac{\log_2 n}{d}$ is assigned label 2^i. For any $x, y \in T$, we denote the *lowest common ancestor* of x and y by $\texttt{LCA}(x, y)$. Let ψ be any arbitrary bijection mapping X to the leaf nodes of T. For any $u, v \in X$, we define $d_X(u, v)$ to be $\Gamma(\texttt{LCA}(\psi(u), \psi(v)))$.

We start with the following basic observation in Claim 9. Although to the best of our knowledge, this precise statement does not appear in previous work, weaker (though similar) versions of this statement can be found in [7, 21].

▷ Claim 9. (X, d_X) is an ultrametric space of doubling dimension d.

Proof. First, we argue that (X, d_X) is an ultrametric space. By the definition of d_X, we have $d_X(u, v) = 0$ iff $u = v$. Further, $d_X(u, v) = d_X(v, u)$ holds for any $u, v \in X$ since $\texttt{LCA}(\psi(u), \psi(v)) = \texttt{LCA}(\psi(v), \psi(u))$. It remains to show $d_X(u, v) \leq \max\{d_X(u, w), d_X(w, v)\}$, for any $u, v, w \in X$. Let T_u (T_v) denote the subtree rooted at the child node of $\texttt{LCA}(\psi(v), \psi(u))$ that contains $\psi(u)$ $(\psi(v))$. Note that for any $w \in X$, $\psi(w)$ cannot belong to both T_u and T_v, so either $\texttt{LCA}(\psi(v), \psi(w))$ is not contained in T_v, or $\texttt{LCA}(\psi(u), \psi(w))$ is not contained in T_u. Thus, $d_X(u, v) \leq \max\{d_X(u, w), d_X(w, v)\}$ holds.

Second, we prove that (X, d_X) has doubling dimension d.

We first show that the doubling dimension is no smaller than d. Let $B(u, r)$ be a ball centered at $u \in X$ with radius $r \in \mathbb{R}^+$. Let $\psi(f_u)$ be the parent of $\psi(u)$ in T. Consider $r = 2$. By the construction of T, every point $v \in X$ corresponding to a child of $\psi(f_u)$ satisfies $d_X(u, v) = \Gamma(\psi(u), \psi(v))$. Hence, $B(u, 2)$ contains all points corresponding to the children of $\psi(f_u)$, and contains at least 2^d distinct points. On the other hand, for any $v \in X$, the ball $B(v, 1)$ contains only the single point v. Hence, covering $B(u, 2)$ requires at least 2^d balls of radius 1. This implies that the doubling dimension is no smaller than d.

Then, we show that the doubling dimension of (X, d_X) is no bigger than d. When $r < 2$, $B(u, r)$ contains only one point u and is covered by $B(u, \frac{r}{2})$. When $2^i \leq r < 2^{i+1}$ for some integer $i \geq 1$, let ω be the ancestor of $\psi(u)$ of height i in T and let T_ω be the subtree rooted at ω. By the construction of T, all pairwise distances between the points corresponding to the leaf nodes of T_ω are no larger than the label 2^i of ω. On the other hand, the distance between u and any point corresponding to a leaf node outside T_ω is no smaller than the label 2^{i+1} of the parent node of ω, as their lowest common ancestor is outside T_ω. Thus, $B(u, r)$ consists of the $(2^d)^i = 2^{d \cdot i}$ points that correspond to the leaf nodes of T_ω. Let L_j be the set of points corresponding to the leaves of the subtree rooted at the jth child of ω, for each $j \in [2^d]$, and let v_j be an arbitrary point in L_j. Since all children of ω have label 2^{i-1}, all pairwise distances between the points in L_j are no larger than 2^{i-1}. So $B(v_j, \frac{r}{2}) \supseteq B(v_j, 2^{i-1})$ covers all $(2^d)^{i-1} = 2^{d \cdot (i-1)}$ points in L_j, for each $j \in [2^d]$, implying that the ball $B(u, r)$ can be covered by the 2^d balls $B(v_j, \frac{r}{2}), j \in [2^d]$. ◁

Then, we demonstrate the following property of (X, d_X) in Claim 10 that will be crucial for proving the size and lightness lower bounds.

▷ Claim 10. Let $G = (X, E, w)$ be an arbitrary $(1 + \varepsilon)$-spanner for (X, d_X). For any point pair $u, v \in X$ such that $d_X(u, v) < \frac{2}{\varepsilon}$, edge (u, v) must be in G.

Proof. Consider any pair $u, v \in X$. We first argue that if edge (u, v) is not in G, then $d_G(u, v) \geq d_X(u, v) + 2$. In this case, consider a shortest path from u to v in G, and note that it traverses an intermediate point $w \in X$. By the triangle inequality, $d_G(u, v) \geq d_G(u, w) + d_G(w, v) \geq d_X(u, w) + d_X(w, v)$. Since the minimum pairwise distance is 2 and (X, d_X) is an ultrametric, $d_G(u, v) \geq d_X(u, w) + d_X(w, v) = \max\{d_X(u, w), d_X(w, v)\} + \min\{d_X(u, w), d_X(w, v)\} \geq d_X(u, v) + 2$.

We conclude that if $d_X(u, v) < \frac{2}{\varepsilon}$, then edge (u, v) must be in G, otherwise by the above assertion we would get $d_G(u, v) \geq d_X(u, v) + 2 > (1 + \varepsilon) d_X(u, v)$, which is a contradiction to G being a $(1 + \varepsilon)$-spanner for (X, d_X). ◁

In the following subsections, we apply Claim 10 to prove the size and lightness lower bounds of any $(1 + \varepsilon)$-spanner $G = (X, E, w)$ for (X, d_X).

Size lower bound. Let $D_{\max}$ be the largest label among nodes in T that is strictly smaller than $\frac{2}{\varepsilon}$. Since the labels in T grow geometrically by a factor of 2, there exists a label in $\left[\frac{1}{\varepsilon}, \frac{2}{\varepsilon}\right)$, hence $\frac{1}{\varepsilon} \leq D_{\max} < \frac{2}{\varepsilon}$. Let $i_{\max} = \log(D_{max}) \geq \log(\frac{1}{\varepsilon})$ be the height of nodes with label D_{max}. Each subtree of height $i_{\max}$ has $(2^d)^{i_{\max}} \geq \varepsilon^{-d}$ leaf nodes (this is where the restriction $\varepsilon^{-d} = O(n)$ comes into play) and the pairwise distances between the corresponding points are bounded by $D_{max} < \frac{2}{\varepsilon}$, hence Claim 10 implies that the spanner G connects all these points by a clique. Thus each point in X has a degree of at least $\varepsilon^{-d} - 1$ in G, and so G contains at least $n(\varepsilon^{-d} - 1)/2 = \Omega(n \cdot \varepsilon^{-d})$ edges, which proves the size lower bound.

Lightness lower bound. We begin by giving the lower bound on the weight of G. Consider an arbitrary point $u \in X$ and denote the subtree rooted at the ancestor of $\psi(u)$ at level $i_{\max}$ as $T_{i_{\max}}(u)$. We have shown that u is connected to all $(2^d)^{i_{\max}} \geq \varepsilon^{-d}$ points corresponding to the leaf nodes in $T_{i_{\max}}(u)$. Note also that only $(2^d)^{i_{\max}-1}$ points among those may belong to the child subtree of $T_{i_{\max}}(u)$ that contains $\psi(u)$, while the remaining $(2^d)^{i_{\max}} - (2^d)^{i_{\max}-1} \geq \varepsilon^{-d}/2$ points belong to other child subtrees of $T_{i_{\max}}(u)$, and are therefore at distance $D_{max} = 2^{i_{\max}} \geq 1/\varepsilon$ from u. Thus the total weight of edges incident on any point $u \in X$ in G is at least $\varepsilon^{-d}/2 \cdot (1/\varepsilon) = \varepsilon^{-(d+1)}/2$, and so the weight of G is at least $n \cdot \varepsilon^{-(d+1)}/4 = \Omega(n \cdot \varepsilon^{-(d+1)})$.

Claim 11 concludes the proof of the lightness lower bound for $d \geq 2$. As for $d = 1$, Claim 11 is not enough, since we need $w(\mathrm{MST}_X)$ to be $O(n \cdot \log(\varepsilon^{-1}))$ rather than $O(n \log n)$ to obtain the required lightness bound; we will handle this issue after the proof of Claim 11.

▷ Claim 11. $w(\mathrm{MST}_X) = O(n)$ for any $d \geq 2$ and $w(\mathrm{MST}_X) = O(n \log n)$ for $d = 1$.

Proof. Let P_X be the Hamiltonian path of X that traverses the points of X according to the induced left-to-right ordering in T of the corresponding leaves. To prove the claim, we will show that $w(P_X) = O(n)$ for any $d \geq 2$ and $w(P_X) = O(n \log n)$ for $d = 1$. (It can be shown that $P_X = \mathrm{MST}_X$, but there is no need for that, since we only need to upper bound $w(\mathrm{MST}_X)$ and we have $w(P_X) \geq w(\mathrm{MST}_X)$.)

Denote the root of T by r, and note that $\Gamma(r) = 2^{\frac{\log_2 n}{d}} = n^{1/d}$. Observe that the number of edges in P of weight $\Gamma(r) = n^{1/d}$ is exactly one less than the number of children, i.e., $2^d - 1$. In general, note that the number of edges in P of weight $\frac{\Gamma(r)}{2^i} = \frac{n^{1/d}}{2^i}$ is $2^{d \cdot i}(2^d - 1)$, for each $i \in [0, \frac{\log_2 n}{d} - 1]$. It follows that

$$w(P_X) = \sum_{i=0}^{\frac{\log_2 n}{d}-1} 2^{d \cdot i}(2^d - 1) \cdot \frac{n^{1/d}}{2^i} = (2^d - 1) \cdot n^{1/d} \cdot \sum_{i=0}^{\frac{\log_2 n}{d}-1} (2^{d-1})^i.$$

If $d = 1$, we get that $w(P_X) = n \cdot \log_2 n$.
If $d \geq 2$, then $\sum_{i=0}^{\frac{\log_2 n}{d}-1} (2^{d-1})^i$ is a geometric sum with rate $2^{d-1} \geq 2$, hence we have

$$\sum_{i=0}^{\frac{\log_2 n}{d}-1} (2^{d-1})^i \leq 2 \cdot 2^{(d-1)\cdot(\frac{\log_2 n}{d}-1)} = 2 \cdot 2^{\log_2 n - d - \frac{\log_2 n}{d} + 1} = 4 \cdot \frac{n}{2^d \cdot n^{1/d}},$$

and we get that $w(P_X) = (2^d - 1) \cdot n^{1/d} \cdot \sum_{i=0}^{\frac{\log_2 n}{d}-1} (2^{d-1})^i \leq 4n$. ◁

Finally, consider the case $d = 1$. Claim 11 gives $w(\mathrm{MST}_X) = O(n \log n)$, but to get the required lightness bound we need an upper bound of $w(\mathrm{MST}_X) = O(n \cdot \log(\varepsilon^{-1}))$.

Without loss of generality, we may assume that $n = n'$ and that $n' = \Theta(\varepsilon^{-1})$. In this case, Claim 11 implies that $w(\mathrm{MST}_X) = O(n' \log n') = O(n' \log(\varepsilon^{-1}))$, and the lightness of G is $\Omega(\frac{\varepsilon^{-2}}{\log(\varepsilon^{-1})})$, as required. We introduce two different reductions justifying this assumption.

First, one can simply take the same ultrametric as before on top of n' points (instead of n) as leaves, and then add $n - n'$ points that are arbitrarily close to one of the n' points, to get a metric space with n points and basically the same MST weight; while this reduction provides the required lightness lower bound, it does not preserve the size lower bound.

To get a single instance for which both lower bounds apply, one can do the following. Create n/n' vertex-disjoint copies of the same ultrametric (X, d_X) as before, each on top of n' points as leaves, denoted by $(X^{(j)}, d_{X^{(j)}})$, for each $j \in [n/n']$, and place these copies "on a line", so that neighboring copies $(X^{(j)}, d_{X^{(j)}})$ and $(X^{(j+1)}, d_{X^{(j+1)}})$ are at distance say $2n'$ from each other. Since any two copies are sufficiently spaced apart from each other

(w.r.t. the maximum pairwise distance in each copy, namely n'), any $(1+\varepsilon)$-spanner edge between two points in the same copy must be contained in that copy, hence we can apply the aforementioned size and weight spanner lower bounds on each of the copies separately, and then aggregate their size and weight bounds to achieve the same size and weight lower bounds as before, namely $\Omega(n \cdot \varepsilon^{-1})$ and $\Omega(n \cdot \varepsilon^{-2})$, respectively, for the entire spanner. Since each copy has weight $O(n' \log n')$ by Claim 11 and as neighboring copies are sufficiently close to each other, the MST weight of this metric is bounded by $O(n' \log n') \cdot (n/n') = O(n \cdot \log(\varepsilon^{-1}))$, which proves the required lightness lower bound $\Omega(\frac{\varepsilon^{-2}}{\log(\varepsilon^{-1})})$.

4 Bounded Degree Tree Cover

▶ **Theorem 2.** *For any metric (X, d_X) with doubling dimension d, parameter $0 < \varepsilon < 1$, there exists a $(1+\varepsilon)$-stretch tree cover $\mathcal{T}$ of size $\tilde{O}(\varepsilon^{-d})$, where each tree in $\mathcal{T}$ has maximum vertex degree $O(1)$.*

4.1 Phase 1: Unbounded Degree Tree Cover

▶ **Lemma 12.** *For any metric (X, d_X) with doubling dimension d, and any parameter $0 < \varepsilon < 1$, there is a $(1+\varepsilon)$-stretch tree cover $\mathcal{T}$ of size $O\left(\varepsilon^{-d} \cdot \log(1/\varepsilon)\right)$, where the maximum degree of a vertex in each of the trees in $\mathcal{T}$ is $O(\varepsilon^{-d} \log_{1/\varepsilon} \Delta)$.*

We first give a construction of a tree cover with maximum degree $O(\varepsilon^{-d} \log_{1/\varepsilon} \Delta)$. Later, in Section 4.2, we modify this tree cover to get constant degree bound.

1. **Creating intra-level edge matchings.** Let $\delta_p = 2^p$ for $p \in \{0, \ldots, \lfloor \log(1/\varepsilon) \} \rfloor$, and let τ_{δ_p} be a $(\delta_p, \varepsilon^{-1})$-net tree (see Definition 5) of X and N_i be the $\delta_p \varepsilon^{-i}$-net in τ_{δ_p}. For each p, construct graph G_p, where $V(G_p) = \{u_i \mid u \in N_i\}_{\forall i \in \mathbb{N}}$ and $E(G_p) = \emptyset$. Define a set of *cross edges* $E_c = \{(u_i, v_i) \mid d_X(u,v) \in (\frac{\delta_p}{3\varepsilon^{i+1}}, \frac{4 \cdot \delta_p}{3\varepsilon^{i+1}})\}_{\forall i}$. Partition E_c into $\gamma = O(\varepsilon^{-d})$ matchings $\mathcal{M} = \{M_1, \ldots, M_\gamma\}$, with the property that in each $M \in \mathcal{M}$, $d_X(u_i, v_i) > \frac{14\delta_p}{\varepsilon^i}$, for all matched u_i, v_i. Initialize $T_k^p = (V(G_p), M_k)$ for $k \in \{1, \ldots, \gamma\}$.
2. **Creating inter-level edges.** For each $T = T_k^p$, construct edges between levels i and $(i-1)$ in the order $i = 0, 1, \ldots$. Create a relabel function $\Phi_i^T : N_i \to N_i$, initialized as $\Phi_i^T(u_i) = u_i$ for all $u_i \in N_i$. At each level i, process vertices in the order specified below:
   a. **Matched vertices.** For each matched u_i in M_k, add (u_i, u_{i-1}). For $v \in B_X(u, \frac{4\delta_p}{\varepsilon^i})$ such that v_{i-1} exists, add (u_i, v_{i-1}) if either v_{i-1} is unmatched, or $(v_{i-1}, x_{i-1}) \in M_k$ and x_{i-1} does not have a parent at level i.
   b. **Unmatched vertices.** For each unmatched u_i: enumerate its child nodes in net-tree τ_{δ_p}, denoted by $C_{u_i} = \{a_{i-1}^0, a_{i-1}^1, \ldots, a_{i-1}^\eta\}$. For every $a_{i-1} \in C_{u_i}$:
      - Add $(\Phi_i^T(u_i), a_{i-1})$ in the following cases: 1) a_{i-1} is unmatched and does not have a parent, and 2) $(a_{i-1}, b_{i-1}) \in M_k$ and both a_{i-1}, b_{i-1} do not have parents at level i.
      - Otherwise, if a_{i-1} has a parent x_i, or $(a_{i-1}, b_{i-1}) \in M_k$ and b_{i-1} has a parent x_i, merge x_i with $\Phi_i^T(u_i)$. Set $\Phi_i^T(x_i) = \Phi_i^T(u_i)$ if u_{i+1} exists, otherwise set $\Phi_i^T(u_i) = x_i$. (Note that a_{i-1} and b_{i-1} will not have parents at the same time.)
3. **Contraction.** Contract all vertices sharing label u. $\mathcal{T} = \{T_k^p\}_{p,k}$ forms a tree cover.

Throughout the construction (even after relabeling), the weight of any edge, $w_T(u_i, v_j) = d_X(u, v)$. In Step 2, we slightly abuse the notation, setting $\Phi_i^T(u_i) = x_i$, instead of $\Phi_i^T(\Phi_i^T(u_i)) = x_i$. This is fine, since a node undergoes relabeling at most once. This finishes the construction of the tree cover.

Unless stated otherwise, we refer to the intermediate trees just prior to Step 3 as the tree cover $\mathcal{T}$ and prove our bounds on them. This works since the contraction of only 0-weight edges in Step 3 does not affect the stretch.

We first prove that each T in $\mathcal{T}$, after contraction, is a tree (see Corollary 16). Then, we bound the number of trees in $\mathcal{T}$ (see Lemma 17). Finally, we use Claim 18 and Claim 19 to bound the final stretch in Lemma 20 (proof deferred to the Appendix B of the full version).

We first require the following combinatorial lemma of [21], used in Step 1 of our construction.

▶ **Lemma 13** (Lemma 6 of [21]). *Consider a graph $G = (V, E_b \cup E_r)$ with disjoint sets of blue edges E_b and red edges E_r, such that the maximum blue degree is $\delta_b \geq 1$, and the maximum red degree is $\delta_r \geq 1$. Then there is a set of at most $\gamma = O(\delta_b \delta_r)$ matchings $\mathcal{M} = \{M_1, \ldots, M_\gamma\}$ of G such that:*

1. *$E_b \subseteq \cup_{i=1}^{\gamma} M_i$, and*
2. *for every matching $M \in \mathcal{M}$, there is no red edge whose both endpoints are matched by M.*

▶ Remark 14. Note that we apply this lemma in Step 1 of the construction. More specifically, $E_b = E_c = \{(u_i, v_i) \mid d_X(u, v) \in (\frac{\delta_p}{3\varepsilon^{i+1}}, \frac{4\delta_p}{3\varepsilon^{i+1}})\}_{\forall i}$, and $E_r = \{(u_i, v_i) \mid d_X(u, v) \leq \frac{14\delta_p}{\varepsilon^i}\}$. In each $M \in \mathcal{M}$, $d_X(u_i, v_i) > \frac{14\delta_p}{\varepsilon^i}$ holds for all matched u_i, v_i.

We begin by proving that each $T_k^p \in \mathcal{T}$ is a tree. We rely on the following inductive invariant: after processing the i^{th} level, every level-i subtree has a diameter at most $O(\varepsilon)$ factor of the newly added level-i cross edge. This large separation ensures that the forest remains a forest.

▶ **Lemma 15.** *Each $T_k^p \in \mathcal{T}$ is a tree.*

Proof. Let $T = T_k^p$. We will prove the following two properties. First, each vertex at level-i is connected to at most one vertex at level-$(i+1)$. Second, for every level-i cross edge, at most one of the end points is connected to a level-$(i+1)$ vertex. These two arguments are sufficient condition for T to be acyclic since all cross edges are mutually vertex-disjoint.

We begin by proving that these two properties hold in Step 2a. For any vertex $v_{i-1} \in B_X(u, \frac{4\delta_p}{\varepsilon^i})$ and u_i is a matched vertex of T, v_i cannot also be a matched vertex of T since $d_X(u, v) > \frac{14\delta_p}{\varepsilon^i}$ should hold by Lemma 13. Hence, any vertex v_{i-1} is connected to at most one vertex at level-$(i+1)$. For any matched cross edge $(u_{i-1}, v_{i-1}) \in T$, we have $d_X(u, v) \in (\frac{\delta_p}{3\varepsilon^i}, \frac{4\delta_p}{3\varepsilon^i})$, in which case u_i and v_i cannot both be matched vertices in level-i since this requires $d_X(u, v) > \frac{14\delta_p}{\varepsilon^i}$. Thus, (u_i, u_{i-1}) and (v_i, v_{i-1}) will not be added at the same time, and (u_{i-1}, v_{i-1}) cannot have parents at both endpoints. Next, by construction (Step 2b), the two properties hold. This is because an edge between some a_{i+1} and b_i is created only when either b_i is unmatched and does not have a parent, or some (b_i, c_i) exists and neither of b_i or c_i have a parent. At each level of the net tree τ_{δ_p}, we go over all vertices.

Let rt be the vertex at the highest level of T. For any two vertices $u, v \in V(T)$, we know that there is a path from u to rt, and from v to rt in T. This holds because each vertex is either connected to a vertex one level above, or to another vertex at the same level via a cross edge (since each such vertex is enumerated in Step 2b). Hence, there is a walk between u and v via rt in T. This implies that T is connected. ◀

▶ **Corollary 16.** ($\star$) *After the contraction procedure, each T_k^p in $\mathcal{T}$ remains a tree.*

▶ **Lemma 17.** *The number of trees in $\mathcal{T}$ is $O(\varepsilon^{-d} \log(1/\varepsilon))$.*

Proof. For each G_p ($p \in \{0, 1 \ldots, \lfloor \log(1/\varepsilon) \rfloor\}$, we take $V = V(G_p)$, $E_b = E_c$. We define a set of dummy edges in G_p: $E_r = \{(u_i, v_i) \mid d_X(u, v) \leq \frac{14\delta_p}{\varepsilon^i}\}_{\forall i}$. By packing lemma, the maximum degree in E_b at each level i is $O((\delta_p\varepsilon^{-(i+1)}/\delta_p\varepsilon^{-i})^d) = O(\varepsilon^{-d})$.

Similarly, the maximum degree in E_r is $O(14\delta_p\varepsilon^{-i}/\delta_p\varepsilon^{-i})^d = c^d$ for some constant c. We can now invoke Lemma 13 on $G = (V, E_b \cup E_r)$ with $\delta_b = O(\varepsilon^{-d})$ and $\delta_r = c^d$ to get $O((c\varepsilon^{-1})^d) = O(\varepsilon^{-d})$ matchings (each corresponding to a tree) with the required properties. Hence, the total number of trees in $\mathcal{T}$ is $O(\gamma \cdot \log(1/\varepsilon)) = O(\varepsilon^{-d} \log(1/\varepsilon))$. ◀

▷ Claim 18. ($\star$) Let $T = T_k^p$. Let (a_i, b_i) be a cross edge and (c_i, d_{i-1}) be an inter-level edge in T. For $\varepsilon < 1/156$, we have

$$\frac{\delta_p}{4\varepsilon^{i+1}} \leq d_T(a_i, b_i) \leq \frac{3\delta_p}{2\varepsilon^{i+1}}, \quad \text{and} \quad d_T(c_i, d_{i-1}) \leq \frac{27}{2}\frac{\delta_p}{\varepsilon^i}.$$

▷ Claim 19. ($\star$) For any $T_k^p \in \mathcal{T}$, the distance from a leaf node u_0 to any of its ancestors at level i is at most $2\delta_p\varepsilon^{-(i+1)}$.

▶ **Lemma 20.** ($\star$) *For every $u, v \in X$, there exists a tree T in the tree cover $\mathcal{T}$ such that: $d_T(u, v) \leq (1 + \varepsilon)d_X(u, v)$.*

4.2 Phase 2: Bounding the Degree

The algorithm in Section 4.1 builds the tree cover $\mathcal{T}$ of size $O(\varepsilon^{-d} \log(1/\varepsilon))$. In this section, we focus on adjusting edges of these trees to achieve constant degree bound. We state the main algorithm and refer to the Appendix B of the full version for the detailed analysis of this section.

Let T denote a tree in the tree cover $\mathcal{T}$ constructed before Step 3 (contraction). Let $l(x)$ be the highest level of x in the net tree τ_{δ_p}. We assign a direction to each edge of T based on the levels of its endpoints in τ_{δ_p}. An edge $e = (a_i \to b_j)$ is going out from a_i to b_j if $l(a) \leq l(b)$. If $l(a) = l(b)$, we assign the direction arbitrarily. Similar to Phase 1, we categorize the set of directed edges into two categories:

1. An edge $(x_i \to p_{i+1})$ is an *inter-level edge at level i* if $x \in N_i \setminus N_{i+1}$ and $p \in N_{i+1}$. Let D_i be the set of inter-level edges at level i. We denote the set of points p_{i+1} incident to an inter-level edge $(x_i \to p_{i+1})$ at level i by $D_i^-(x)$, and the set of points w_i incident to an inter-level edge $(w_i \to x_{i+1})$ at level i by $D_i^+(x)$.
2. An edge $(x_i \to y_i)$ is a *cross edge at level i* if $x, y \in N_i$. Let C_i be the set of all cross edges at level i. Given a point x_i, we denote the set of all points y_i having a cross edge $(y_i \to x_i)$ by $C_i^+(x)$, and the set of all points z_i having a cross edge $(x_i \to z_i)$ by $C_i^-(x)$.

By our Phase 1 construction, we observe the following bounds:

▶ **Observation 21.** *Given a fixed level i, for every vertex $x \in N_i$, $|D_i^+(x)| = O(\varepsilon^{-d})$, $|D_i^-(x)| \leq 1$, $|C_i^+(x)| \leq 1$, $|C_i^-(x)| \leq 1$. Furthermore, for every point x, we have $\sum_i |D_i^-(x)| \leq 1$, and $\sum_i |C_i^-(x)| \leq 2$.*

We use the degree reduction technique of [8] to reduce degree associated with both cross edges and inter-level edges. The input and output of this technique are specified in Remark 22 and Lemma 23, respectively.

▶ Remark 22. The input tree T must be a tree in the tree cover, and we consider a subset E of edges in T. Let E_i denote the edges in E at level i, and $E_i^+(x)$ (resp., $E_i^-(x)$) denote the set of points connected to x by incoming (resp., outgoing) edges in E_i. The subset E must satisfy the following degree constraints for every point x: 1) $O(1)$ in-coming edges at every level, 2) at most one out-going edge at every level, and 3) totally $O(1)$ out-going edges over all levels. Formally, $|E_i^+(x)| \le d_{in}$, $|E_i^-(x)| \le 1$ and $\sum_i |E_i^-(x)| \le d_{out}$, where $d_{in} > 0$ and $d_{out} > 0$ are constants.

▶ **Lemma 23.** *Given a tree cover $\mathcal{T}$, let T be a tree in $\mathcal{T}$ that satisfies Remark 22. There exists an algorithm redirecting edges in E of T to obtain a tree T' with a new edge set E' such that:*

- *(Bounded degree.) Every point has at most $2 + 2d_{in} + d_{out}$ edges in E'.*
- *(Small stretch.) For every edge (y, x) in E of T, there exist a path π in T' from y to x such that π contains edges in E', and $d_{T'}(y, x) \le (1 + O(\varepsilon)) \cdot d_X(y, x)$.*

Algorithm. We use two operations: redirect cross edges and redirect inter-level edges.

- **Step 1. Redirecting cross edges.** We apply Lemma 23 to re-direct cross edges, where $E = \bigcup_i C_i$. E_i^- and E_i^+ in Remark 22 will respectively be C_i^- and C_i^+, where $|C_i^+(x)| \le 1, |C_i^-(x)| \le 1, \sum_t |C_t^-(x)| \le 2$ for every point x and every level i (by Observation 21).
- **Step 2. Redirecting inter-level edges.**
  - **Step 2.1.** We visit levels from the bottom to top. Let Q_i be the set of new edges at level i created by this step. Initialize $Q_i = \emptyset$. Let $Q_i^-(p)$ be the set of points q having an edge $(p \to q)$ in Q_i. Let $Q_i^+(p)$ be the set of points q having an edge $(q \to p)$ in Q_i. Given a fixed level i, for every vertex $x_{i+1} \in N_{i+1}$, recall that the set of in-coming inter-level edges at level i of x is $D_i^+(x) = \{w_i^{(1)}, w_i^{(2)}, \ldots\}$, we know $|D_i^+(x)| \le O(\varepsilon^{-d})$. We sort D_i^+ by increasing order of distances to x_{i+1}. Consider $w_i^{(j)} \in D_i^+(x)$:
    - If $j \le 2$: keep $(w_i^{(j)} \to x)$.
    - If $j > 2$: let $j^* = \lfloor (j-1)/2 \rfloor$; remove $(w_i^{(j)} \to x_{i+1})$ and create edge $(w_i^{(j^*)} \to w_i^{(j)})$ with weight $d_X(w_i^{(j^*)}, w_i^{(j)})$. We add $(w_i^{(j^*)} \to w_i^{(j)})$ to Q_i.
  - **Step 2.2.** We apply Lemma 23 to re-direct inter-level edges remaining after Step 2.1, where $E = \bigcup_i D_i$. E_i^- and E_i^+ in Remark 22 will be D_i^- and D_i^+ respectively, where $|D_k^+(p)| \le 2, |D_k^-(p)| \le 1, \sum_t |D_t^-(p)| \le 1$ for every point p and level k. These bounds hold by Observation 21 and by Step 2.1 (when $j \le 2$).
  - **Step 2.3.** We apply Lemma 23 to re-direct $E = \bigcup_i Q_i$ created by Step 2.1. E_i^- and E_i^+ in Remark 22 will be respectively Q_i^- and Q_i^+.

This finishes the construction of the bounded degree tree cover.

References

1 P. K. Agarwal, Y. Wang, and P. Yin. Lower bound for sparse Euclidean spanners. In *Proc. of 16th SODA*, pages 670–671, 2005.

2 I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Computational Geometry*, 9(1):81–100, 1993. doi:10.1007/BF02189308.

3 S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: Short, thin, and lanky. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 489–498, 1995.

4 S. Arya and M. H. M. Smid. Efficient construction of a bounded degree spanner with low weight. *Algorithmica*, 17(1):33–54, 1997. doi:10.1007/BF02523237.

5 Yair Bartal, Arnold Filtser, and Ofer Neiman. On notions of distortion and an almost minimum spanning tree with constant average distortion. *J. Comput. Syst. Sci.*, 105:116–129, 2019. preliminary version published in SODA 2016. doi:10.1016/j.jcss.2019.04.006.

6 G. Borradaile, H. Le, and C. Wulff-Nilsen. Greedy spanners are optimal in doubling metrics. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 2371–2379, 2019. doi:10.1137/1.9781611975482.145.

7 H. T.-H. Chan and A. Gupta. Small hop-diameter sparse spanners for doubling metrics. In *Proc. of 17th SODA*, pages 70–78, 2006.

8 T-H. Hubert Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'05, 2005.

9 T.-H. Hubert Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics. *ACM Trans. Algorithms*, 12(4):55:1–55:22, 2016. Preliminary version appeared in SODA 2005. doi:10.1145/2915183.

10 T.-H. Hubert Chan, Mingfei Li, and Li Ning. Sparse fault-tolerant spanners for doubling metrics with bounded hop-diameter or degree. In *ICALP (1)*, pages 182–193, 2012. Full version in Algorithmica'15. doi:10.1007/978-3-642-31594-7_16.

11 T.-H. Hubert Chan, Mingfei Li, Li Ning, and Shay Solomon. New doubling spanners: Better and simpler. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Proc. 40th ICALP*, volume 7965 of *Lecture Notes in Computer Science*, pages 315–327. Springer, 2013. Full version in SICOMP'15. doi:10.1007/978-3-642-39206-1_27.

12 B. Chandra, G. Das, G. Narasimhan, and J. Soares. New sparseness results on graph spanners. In *Proceedings of the Eighth Annual Symposium on Computational Geometry*, 1992.

13 Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenković, Shay Solomon, and Cuong Than. Optimal Euclidean Tree Covers. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.37.

14 L. P. Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, pages 169–177, 1986.

15 K. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 56–65, 1987.

16 G. Das, P. Heffernan, and G. Narasimhan. Optimally sparse spanners in 3-dimensional euclidean space. In *Proceedings of the 9th Annual Symposium on Computational Geometry*, SCG '93, pages 53–62, 1993.

17 G. Das and G. Narasimhan. A fast algorithm for constructing sparse Euclidean spanners. In *Proc. of 10th SOCG*, pages 132–139, 1994.

18 G. Das, G. Narasimhan, and J. Salowe. A new way to weigh malnourished euclidean graphs. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '95, pages 215–222, 1995.

19 Y. Dinitz, M. Elkin, and S. Solomon. Shallow-low-light trees, and tight lower bounds for Euclidean spanners. In *Proc. of 49th FOCS*, pages 519–528, 2008.

20 Michael Elkin and Shay Solomon. Optimal euclidean spanners: Really short, thin, and lanky. *J. ACM*, 62(5):35:1–35:45, 2015. doi:10.1145/2819008.

21 Arnold Filtser and Hung Le. Locality-sensitive orderings and applications to reliable spanners. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC '22, pages 1066–1079. ACM, 2022. doi:10.1145/3519935.3520042.

22 Jie Gao, Leonidas J. Guibas, and An Nguyen. Deformable spanners and applications. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 190–199, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/997817.997848.

23 L. Gottlieb and L. Roditty. Improved algorithms for fully dynamic geometric spanners and geometric routing. In *Proc. of 19th SODA*, pages 591–600, 2008.

24 L. Gottlieb and L. Roditty. An optimal dynamic spanner for doubling metric spaces. In *Proc. of 16th ESA*, pages 478–489, 2008. Another version of this paper is available via `http://cs.nyu.edu/~adi/spanner2.pdf`.

25 Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. Efficient regression in metric spaces via approximate lipschitz extension. *IEEE Trans. Inf. Theory*, 63(8):4838–4849, 2017. `doi:10.1109/TIT.2017.2713820`.

26 J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.*, 31(5):1479–1500, 2002. `doi:10.1137/S0097539700382947`.

27 J. Gudmundsson, C. Levcopoulos, G. Narasimhan, and M. H. M. Smid. Approximate distance oracles for geometric graphs. In *Proc. of 13th SODA*, pages 828–837, 2002.

28 J. Gudmundsson, C. Levcopoulos, G. Narasimhan, and M. H. M. Smid. Approximate distance oracles for geometric spanners. *ACM Transactions on Algorithms*, 4(1), 2008. `doi:10.1145/1328911.1328921`.

29 J. Gudmundsson, G. Narasimhan, and M. H. M. Smid. Fast pruning of geometric spanners. In *Proc. of 22nd STACS*, pages 508–520, 2005.

30 Joachim Gudmundsson, Christos Levcopoulos, Giri Narasimhan, and Michiel H. M. Smid. Approximate distance oracles revisited. In *Proc. of 13th ISAAC*, pages 357–368, 2002. `doi:10.1007/3-540-36136-7_32`.

31 S Har-Peled and M Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006. `doi:10.1137/s0097539704446281`.

32 Y. Hassin and D. Peleg. Sparse communication networks and efficient routing in the plane. In *Proc. of 19th PODC*, pages 41–50, 2000.

33 Omri Kahalon, Hung Le, Lazar Milenkovic, and Shay Solomon. Can't see the forest for the trees: Navigating metric spaces by bounded hop-diameter spanners. In Alessia Milani and Philipp Woelfel, editors, *Proc. PODC*, pages 151–162. ACM, 2022. `doi:10.1145/3519270.3538414`.

34 J. M. Keil. Approximating the complete euclidean graph. In *Proceedings of the first Scandinavian Workshop on Algorithm Theory*, SWAT '88, pages 208–213, 1988.

35 J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete and Computational Geometry*, 7(1):13–28, 1992. `doi:10.1007/BF02187821`.

36 An La and Hung Le. Dynamic locality sensitive orderings in doubling metrics. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, STOC '25, pages 1178–1189, New York, NY, USA, 2025. Association for Computing Machinery. `doi:10.1145/3717823.3718209`.

37 H. Le and S. Solomon. Near-optimal spanners for general graphs in (nearly) linear time. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '22, pages 3332–3361. 2022. `doi:10.1137/1.9781611977073.132`.

38 Hung Le and Shay Solomon. Truly optimal euclidean spanners. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1078–1100. IEEE Computer Society, 2019. `doi:10.1109/FOCS.2019.00069`.

39 Hung Le and Shay Solomon. A unified framework for light spanners. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 295–308. ACM, 2023. `doi:10.1145/3564246.3585185`.

40 Hung Le, Shay Solomon, and Cuong Than. Optimal fault-tolerant spanners in euclidean and doubling metrics: Breaking the ω (log n) lightness barrier. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 77–97. IEEE, 2023. `doi:10.1109/FOCS57990.2023.00013`.

41 Hung Le and Cuong Than. Greedy spanners in euclidean spaces admit sublinear separators. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proc. SODA*, pages 3287–3310. SIAM, 2022. doi:10.1137/1.9781611977073.130.

42 Y. Mansour and D. Peleg. An approximation algorithm for min-cost network design. *DIMACS Series in Discr. Math and TCS*, 53:97–106, 2000.

43 G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

44 S. B. Rao and W. D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, STOC '98, pages 540–550, 1998. Full version at http://graphics.stanford.edu/courses/cs468-06-winter/Papers/rs-tsp.pdf. doi:10.1145/276698.276868.

45 J. Ruppert and R. Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proceedings of the 3rd Canadian Conference on Computational Geometry*, CCCG '91, pages 207–210, 1991.

46 Jeffrey S. Salowe. On euclidean spanner graphs with small degree. In *Proceedings of the Eighth Annual Symposium on Computational Geometry, Berlin, Germany, June 10-12, 1992*, pages 186–191, 1992. doi:10.1145/142675.142716.

47 M. Smid. The weak gap property in metric spaces of bounded doubling dimension. In Susanne Albers, Helmut Alt, and Stefan Näher, editors, *Efficient Algorithms*, pages 275–289. Springer-Verlag, 2009.

48 Shay Solomon. From hierarchical partitions to hierarchical covers: optimal fault-tolerant spanners for doubling metrics. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 363–372, 2014. doi:10.1145/2591796.2591864.

49 A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982. doi:10.1137/0211059.

Approximate Dynamic Nearest Neighbor Searching in a Polygonal Domain

Joost van der Laan
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Frank Staals
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Lorenzo Theunissen
Delft University of Technology, The Netherlands

Abstract

We present efficient data structures for approximate nearest neighbor searching and approximate 2-point shortest path queries in a two-dimensional polygonal domain P with n vertices. Our goal is to store a dynamic set of m point sites S in P so that we can efficiently find a site $s \in S$ closest to an arbitrary query point q. We will allow both insertions and deletions in the set of sites S. However, as even just computing the distance between an arbitrary pair of points $q, s \in P$ requires a substantial amount of space, we allow for approximating the distances. Given a parameter $\varepsilon > 0$, we build an $O(\frac{n}{\varepsilon} \log n)$ space data structure that can compute a $1+\varepsilon$-approximation of the distance between q and s in $O(\frac{1}{\varepsilon^2} \log n)$ time. Building on this, we then obtain an $O(\frac{n+m}{\varepsilon} \log n + \frac{m}{\varepsilon} \log m)$ space data structure that allows us to report a site $s \in S$ so that the distance between query point q and s is at most $(1+\varepsilon)$-times the distance between q and its true nearest neighbor in $O(\frac{1}{\varepsilon^2} \log n + \frac{1}{\varepsilon} \log n \log m + \frac{1}{\varepsilon} \log^2 m)$ time. Our data structure supports updates in $O(\frac{1}{\varepsilon^2} \log n + \frac{1}{\varepsilon} \log n \log m + \frac{1}{\varepsilon} \log^2 m)$ amortized time.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases dynamic data structure, nearest neighbor search, polygonal domain

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.69

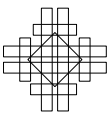
Related Version *Full Version*: `https://arxiv.org/abs/2603.11775` [20]

1 Introduction

Nearest neighbor searching is a fundamental problem in which the goal is to store a set S of m point sites so that given a query point q one can quickly report a site in S that is closest to q. For example, the set S may be a set of (locations of) shops, and we may wish to efficiently find the shop closest to our current location. In addition, nearest neighbor searching shows up as an important subroutine in several other fundamental problems. For example, in computing closest pairs [7] and matchings [15]. In these applications, there are often two (additional) difficulties: (i) the point sites are typically embedded in some constrained domain, and (ii) the set of sites may change over time, and hence it may be necessary to insert and delete sites. In the example problem above other buildings, roads, and lakes etc. act as obstacles, and thus we actually have to measure the distance $d(q, s)$ between two points q and s in terms of the length of a shortest obstacle avoiding path $\pi(q, s)$. Furthermore, shops may open and close (or may be too crowded or out of products etc.), and hence we want to consider only the currently open or available shops when querying.

In this paper, we consider dynamic nearest neighbor searching among a set S of m sites in a two-dimensional polygonal domain P with n vertices. We will allow both insertions and deletions in the set of sites S. Agarwal, Arge, and, Staals [1] presented a near linear space data structure for this problem that allows efficient (polylogarithmic) queries and updates, provided that P contains no holes; i.e. it is a simple polygon. However, extending these


© Joost van der Laan, Frank Staals, and Lorenzo Theunissen;
licensed under Creative Commons License CC-BY 4.0



results to arbitrary polygonal domains seems to be far out of reach. It seems we need to at least be able to efficiently compute the (geodesic) distance $d(q, s)$ between two arbitrary query points q and s in P. Currently, the best data structures that support such queries in polylogarithmic time use $O(n^9)$ space [11] (and it seems unlikely we can determine the site closest to q without actually evaluating geodesic distances). Hence, we settle on answering queries approximately. Let $\varepsilon > 0$ be some parameter, and let s^* be the site in S closest to a query point q, then our goal is to report a site $s \in S$ whose distance to q is at most $(1 + \varepsilon)d(q, s^*)$. We will say that s is *ε-close* to q. Our main result is then:

▶ **Theorem 1.** *Let P be a polygonal domain with n vertices, let S be a dynamic set of m point sites in P, and let $\varepsilon > 0$ be a parameter. There is an $O(\frac{n+m}{\varepsilon} \log n + \frac{m}{\varepsilon} \log m)$ space data structure that can*

- *compute a site in S that is ε-close to a query $q \in P$ in $O(\frac{1}{\varepsilon^2} \log n + \frac{1}{\varepsilon} \log n \log m + \frac{1}{\varepsilon} \log^2 m)$ time,*
- *support insertions of sites into S in $O(\frac{1}{\varepsilon^2} \log n + \frac{1}{\varepsilon} \log n \log m + \frac{1}{\varepsilon} \log^2 m)$ amortized time,*
- *and support deletions of sites from S in $O(\frac{1}{\varepsilon} \log n \log m + \frac{1}{\varepsilon} \log^2 m)$ amortized time.*

Constructing the initially empty data structure takes $O(\frac{n}{\varepsilon^2} \log^2 n)$ time.

The crucial ingredient in our data structure is a data structure for $(1 + \varepsilon)$-approximate 2-point shortest path queries. Thorup [19] presents an $O(\frac{n}{\varepsilon} \log n)$ space data structure that, given two arbitrary query points s, t in the polygonal domain P, can compute an estimate $\hat{d}(s, t)$ of their distance so that $d(s, t) \leq \hat{d}(s, t) \leq (1 + \varepsilon)d(s, t)$ in $O(\frac{1}{\varepsilon^3} + \frac{\log n}{\varepsilon \log \log n})$ time. Thorup's solution however assumes that distances are encoded as floating point numbers whose bits we can manipulate. This is inconsistent with the Real-RAM model of computation usually assumed in computational geometry. We stick to the Real-RAM model, and show that we can still compute approximate distances efficiently. We can actually improve the dependency on ε, and obtain:

▶ **Theorem 2.** *Let P be a polygonal domain with n vertices, and let $\varepsilon > 0$ be a parameter. In $O(\frac{n}{\varepsilon^2} \log^2 n \log \frac{n}{\varepsilon})$ time we can construct an $O(\frac{n}{\varepsilon} \log n)$ space data structure that, given query points $s, t \in P$ returns an $(1 + \varepsilon)$-approximation of $d(s, t)$ in $O(\frac{1}{\varepsilon^2} \log n)$ time.*

Our overall approach is actually the same as in Thorup's approach: we recursively construct a separator, and estimate distances by going via a discrete set of anchor points on this separator. However, our approach provides stronger guarantees on these anchor points. We believe this may be useful for other problems as well. Indeed, it allows us to efficiently answer (approximate) nearest neighbor queries as well. This also still requires several additional ideas, as simply evaluating the distance between a query point and every site would only give us query time linear in m.

Related Work. Dynamic nearest neighbor searching in the Euclidean plane has a long history, with early results by Agarwal and Matoušek [2]. Chan [6] was the first to achieve expected polylogarithmic update and query times. Following a sequence of improvements, the best structure at this time uses $O(m)$ space, allows for $O(\log^2 m)$ time queries, and for updates in deterministic, amortized $O(\log^4 m)$ time [15, 8]. For more general (constant complexity) distance functions, one can similarly achieve expected polylogarithmic query and update times [15, 16]. When the domain is a simple polygon P with n vertices, Oh and Ahn [17] presented a solution using $O(n + m)$ space achieving $O(\sqrt{m} \log(n + m))$ query time, and $O(\sqrt{m} \log m \log^2 n)$ update time. Agarwal, Arge, and, Staals [1] then showed how to achieve $O(\log^2 n \log^2 m)$ query time and expected $O(\text{polylog}(nm))$ update time using

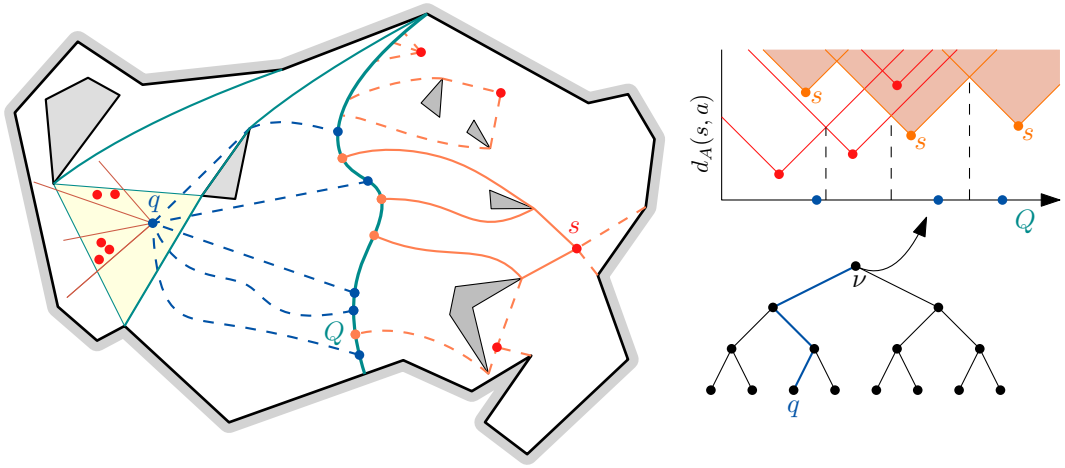


Figure 1 An overview of our approach. We use shortest paths to recursively subdivide P into subpolygons. If s^* lies in the same triangle as q, we can directly compute an ε-close site. Otherwise, the shortest path must cross a shortest path Q, and we find a site s that is sufficiently close to q whose path goes via an anchor point on Q.

$O(n + m \log^3 m \log n)$ space. When the domain may have holes and the set of sites is static, there is an $O(n + m)$ space data structure that allows querying the (exact) nearest neighbor in $O(\log(n + m))$ time [14]. Constructing the data structure uses $O(n \log n)$ time. Wang [21] recently showed that if P is triangulated, the shortest path map of a single site can be computed in $O(n + h \log h)$ time, where h is the number of holes in P. Conceivably, his algorithm could be extended to handle multiple sites (as is the case in the algorithm by Hershberger and Suri). We are not aware of any results that allow updating sites. Even the case where we allow only insertions is challenging: we cannot easily apply a static-to-dynamic transformation [3] since rebuilding the static structure may require $\Omega(n)$ time.

2 Global Approach

We first present an overview of our data structure. See also Figure 1. We start by briefly reviewing the ε-approximate 2-point shortest path query data structure by Thorup [19].

Computing approximate shortest paths. Thorup's approach recursively partitions the domain P using *separators* such that: each separator consists of at most three shortest paths, and the resulting subpolygons are of roughly the same size. This results in a balanced binary tree $\mathcal{T}$ of height $O(\log n)$ in which each node ν corresponds to some subpolygon P_ν that is further subdivided using a separator Q_ν.

Consider the separator Q of the root of this tree $\mathcal{T}$, and let $k = \lceil 1/\varepsilon \rceil$. The main idea is that each polygon vertex v generates a set $A_Q(v)$ of $O(k)$ so called *anchor points*[1] on Q, so that for any point $q \in Q$ the distance via one of these anchor points $A_Q(v)$ to v is at most $(1 + \varepsilon)$ times the true distance $d(q, v)$. So, if a shortest path $\pi(u, v)$ between two vertices u and v intersects Q then there is a path from u to v via two anchor points $a \in A_Q(u)$ and $a' \in A_Q(v)$ (and the subcurve $Q[a, a']$ of Q) of length $(1 + \varepsilon)d(u, v)$. Given the anchor points

[1] Thorup calls these points connections, but we use anchor points to avoid confusion with the paths connecting v to these points.

(and the distances to their defining vertices) we can find the length of such a path in $O(k)$ time. If the path $\pi(u, v)$ does not intersect Q, it is contained in one of the subpolygons of the children of the root, and hence we can compute a $(1 + \varepsilon)$-approximation of the distance $d(u, v)$ in the appropriate subpolygon. As each vertex defines $O(k)$ anchor points on Q, and appears in $O(\log n)$ levels of the recursion. The total space used is $O(nk \log n)$.

To compute the anchor points efficiently, Thorup's actual algorithm uses a graph G in which the distance $d_G(u, v)$ between u and v in the graph is already an $(1+\varepsilon_1)$-approximation of the true distance $d(u, v)$. Using the above approach with parameter ε_0 then actually gives a path in P with total length at most $(1 + \varepsilon_0)(1 + \varepsilon_1)d(u, v)$. Setting $\varepsilon_0 = \varepsilon_1 = \varepsilon/3$ then gives a ε-approximation of the distances as desired.

However, using the graph G results in slightly different (generally weaker) properties compared to the ideal case sketched above. In particular, his approach now guarantees that the distances from vertex v are within a factor $1 + \varepsilon$ of the true distance only for a discrete subset of points on Q. For querying the distances between pairs of vertices this is not an issue. Thorup's data structure can handle such queries in only $O(k)$ time. To support queries between arbitrary points $s, t \in P$, his algorithm connects s and t to $O(k)$ vertices in G each, and then queries the all these pairs. This leads to the query time $O(k^3 + k \log n/ \log \log n)$.

Reducing the query time. Ideally, we would treat s and t identical to polygon vertices, so that we get $O(k)$ anchor points on a separator Q. However, as such a candidate anchor point $p \in Q$ may lie anywhere on Q (in particular, it is not necessarily a vertex of G yet), we may not have the required ε_1-approximation of the distance to such a point yet.

We will show that we can use a slightly different graph G^Q that includes additional "Steiner" points on Q. This leads to a different set of anchor points compared to Thorup's approach, but *does* guarantee that for any vertex v, and *any* point q on Q, the distance to v is at most $(1 + \varepsilon)d(q, v)$. Furthermore, we show that we can efficiently compute an initial set of $O(k^2)$ candidate anchor points for an arbitrary point s, which we then reduce to $O(k)$. This results in $O(k^2 \log n)$ query time for an arbitrary pair of query points $s, t \in P$.

Answering nearest neighbor queries. As in the approximate 2-point shortest path data structure above, we construct a tree $\mathcal{T}$ of separators. We associate every site $s \in S$ with a root-to-leaf path in this tree. Namely, the nodes ν for which s is contained in the subpolygon P_ν. For each node ν on this path, we will additionally maintain a set of anchor points $A_Q(s)$ on the separator $Q = Q_\nu$ corresponding to this node. In particular, we consider Q as a one dimensional space, and maintain an additively weighted Voronoi diagram $Vor(A_Q)$ on Q. The set of sites $A_Q = \bigcup_{s \in S \cap P_\nu} A_Q(s)$ is the set of all anchor points of all sites that appear in P_ν. The weight of an anchor point $a \in A_Q(s)$ is the distance estimate from a to s.

If a shortest path between a query point q and its closest site $s^* \in S$ intersects Q, then there is a sufficiently short path via one of the anchor points of s^* and q on Q. We can then find a sufficiently close site s by querying the Voronoi diagram $Vor(A_Q)$ with all anchor points of q. This leads to an overall query time of $O(k^2 \log n + k \log n \log m + k \log^2 m)$, as we perform such queries in $O(\log n)$ nodes in $\mathcal{T}$.

The graph of the function that expresses the distance from a site $s \in S$ to a point p on Q via a given anchor point $a \in A_Q(s)$ is nicely "V-shaped". Hence, we can maintain $Vor(A_Q)$ by maintaining the lower envelope of these functions. This allows us to insert and remove sites in $O(k \log^2(km))$ time. This leads to an overall update time of $O(k^2 \log n + k \log n \log m + k \log^2 m)$. Our structure uses $O(nk \log n + mk \log n + mk \log m)$ space.

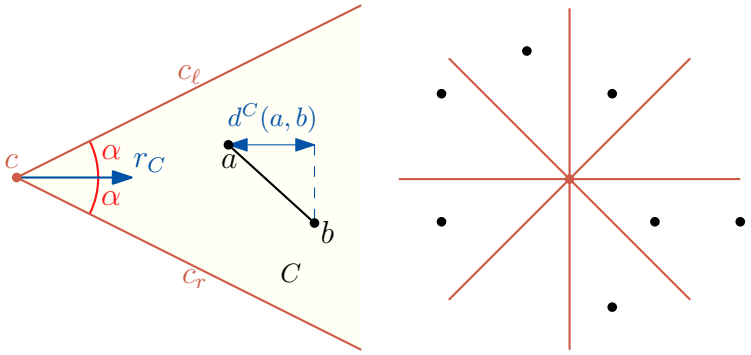


Figure 2 (a) A cone C with apex c, cone direction r_C, and angle α, and an illustration of the cone distance $d^C(a,b)$ between two points $a,b \in C$. (b) A family of cones with angle $2\alpha = 45$ degrees.

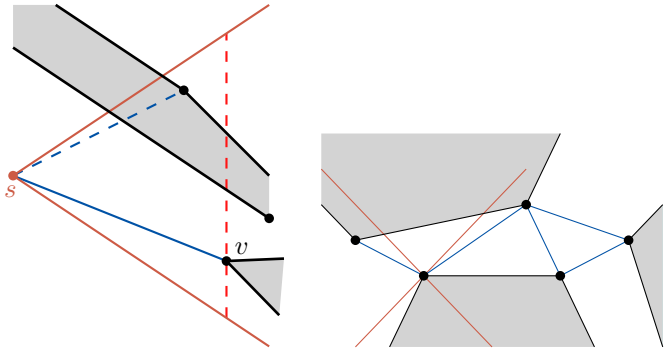


Figure 3 (a) A point s, which has an outgoing cone edge to a vertex v w.r.t the pictured cone C. Hence, there are no visible vertices in C to the left of the red dashed line. (c) An example cone graph G (for a very large ε).

Organization. In Section 3, we define our graph G^Q, and prove that it allows us to accurately estimate distances between any pair of points s,t whose shortest path intersects Q. We then use this to construct our approximate 2-point shortest path data structure in Section 4. Finally, in Section 5 we describe our lower envelope data structure and the remaining tools needed for an efficient dynamic data structure for approximate nearest neighbor searching. Omitted proofs are in the full version [20].

3 Estimating distances using a graph

In this section we discuss how to approximate distances within a polygon P. Our results are based on Clarkson's cone graph [10] on which we can efficiently compute approximate distances between vertices. We review this result in Section 3.1. In Section 3.2 we then extend this approach so that we can represent the distance from any vertex of P to any point on a given shortest path Q in P. This involves computing the anchor points for each vertex of P. Finally, in Section 3.3, we further augment the graph to also represent distances from a set S of arbitrary point sites to (and via) the shortest path Q, and show how to compute the anchor points for such an arbitrary point $s \in S$ efficiently.

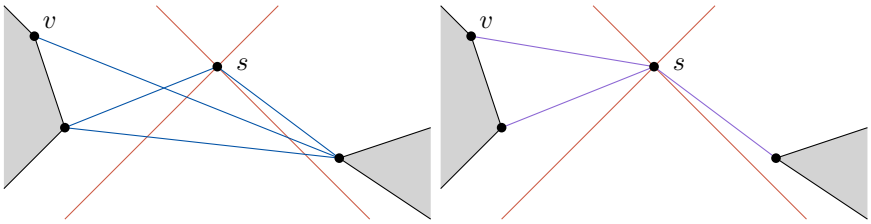


Figure 4 (a) The graph $G[s]$, where only outgoing cone edges from s have been added. (b) The cone graph on $V \cup \{s\}$. Observe that the edge set of neither graph is a subset of the other.

3.1 Clarkson's cone graph

Clarkson [10]'s graph is based on cones. A *cone* C is characterized by an apex point c, a cone direction r_C which is a unit length vector and an angle $\alpha \leq \pi/2$. The boundary of the cone consists of two half lines c_ℓ, c_r who both start at c and the difference in angle compared to r_C is $\alpha/2$ and $-\alpha/2$ respectively. We use C_c to denote a cone C to have its apex at point c. A *cone family* $\mathcal{F}$ is a set of cones that all have their apex at the origin, such that any point in $\mathbb{R}^2$ is contained in at least one cone $C \in \mathcal{F}$. See Figure 2(b) for an example.

Let $\varepsilon > 0$ be a parameter. Clarkson then defines a cone family $\mathcal{F}_\varepsilon$ consisting of $O(k) = O(1/\varepsilon)$ cones, each with angle at most $\varepsilon/8$, and a *cone graph* G based on $\mathcal{F}_\varepsilon$.

Let V be the set of vertices of P. A vertex $b \in V$ is a *minimal cone neighbor* of a point $a \in P$ if and only if there exists a cone $C \in \mathcal{F}_\varepsilon$, so that $b \in S$ is the vertex in the translated cone C_a with minimum *cone-distance* $d^C(a,b) = |r_C \cdot (b-a)|$ to a among all vertices that are visible from a (i.e. for which the line segment $\overline{ab}$ is contained in P). We will also say that a has an *outgoing cone edge* to b w.r.t the cone C. Let $N(a)$ denote the set of these minimal cone neighbors of a over all cones in $\mathcal{F}_\varepsilon$. See Figure 3(a) for an example.

The cone graph $G = (V, E)$ then has an edge $(u, v) \in E$ between two (polygon) vertices $u, v \in V$ if and only if u had v as its *minimal cone neighbor* or vice versa, i.e. $v \in N(u)$ or $u \in N(v)$. Each vertex u has at most $O(k)$ outgoing cone neighbors, and thus the total number of edges in G is at most $O(nk)$. Let $d_G(u, v)$ denote the length of a shortest path $\pi_G(u, v)$ between vertices u, v in G.

▶ **Lemma 3** (Clarkson [10]). *For every pair of vertices u, v in the cone graph G, there is a path $\pi_G(u, v)$ of length $d_G(u, v)$ so that*

$$d(u, v) \leq d_G(u, v) \leq (1 + \varepsilon)d(u, v).$$

In $O(nk \log n)$ time we can build a data structure of size $O(nk)$ so that for any point $s \in P$ one can compute $N(s)$ in $O(k \log n)$ time. Hence, G can be constructed in $O(nk \log n)$ time.

Extending the cone graph. Let $s \in P$ be a point, we define the *extended cone graph* $G[s] = (V \cup \{s\}, E \cup \{(s, v) \mid v \in N(s)\})$ by adding s as a vertex, and connecting it to all of its outgoing cone neighbors. Note that this graph is different from Clarkson's cone graph on $V \cup \{s\}$. See Figure 4 for an illustration. Using essentially same argument as in Lemma 3, we can still prove that $G[s]$ provides an ε-approximation of the distances to and from s.

3.2 Continuous ε-approximation graphs

Let Q be a shortest path between two vertices in P. Our goal is to define and compute, for each vertex v of P, a small set of weighted anchor points $A_Q(v)$ on Q, so that for any point q on Q there is an obstacle avoiding path between q and v of length at most $(1 + \varepsilon)d(q, v)$

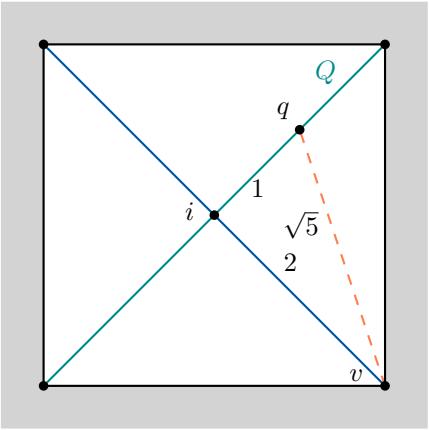


Figure 5 An example that shows that there may be points q on Q (shown in green) for which the graph G' (black, blue and green edges) considered by Thorup does not yield a ε-approximation. For any $\varepsilon \leq 1/10$, Thorup's anchor points give us distance estimate of 3, while the true distance is $\sqrt{5}$. Hence, this is only a $3/\sqrt{5} \geq 1.34$-approximation, not an $(1+\varepsilon)$-approximation.

via one of these anchor points. More specifically, let $\hat{d}_a$ be the weight of an anchor point a. We then want that there is an obstacle avoiding path $\hat{\pi}_a$ from v to a of length $\hat{d}_a$, and that $d_G(v, A_Q(v), q) := \min_{a \in A_Q(v)} d_G(a, q) + \hat{d}_a$ is at most $(1+\varepsilon)d(q, v)$.

Thorup [19] argues that such a set of $O(k)$ anchor points exists. Although his argument (Lemma 5 in his paper) is constructive, it is unclear how to efficiently construct such a set of anchor points. So, he constructs a set of anchor points with respect to Clarkson's cone graph instead. In particular, let $G = (V, E)$ be a cone graph constructed with parameter $\varepsilon_1 = \varepsilon/3$. Thorup defines the graph G' by subdividing every edge in G that intersects Q (at the intersection point), and connecting consecutive vertices along Q, and proves

▶ **Lemma 4** (Thorup [19]). *Let $H = (V, E)$ be an undirected weighted graph, let Q be a shortest path in H, and let $k' \in \mathbb{N}$ be a parameter. For each vertex $u \in H$, we can compute a set $A_Q(u) \subseteq V$ of $O(k')$ discrete anchor points on Q, so that for any vertex $v \in Q$ we have*

$$d_H(u, v) \leq d_H(u, A_Q(u), v) \leq (1 + 1/k')d_H(u, v).$$

Computing all sets of anchor points takes a total of $O\left(k' \log |Q| \left(|V| \log |V| + |E|\right)\right)$ time.

By choosing $1/k' = \varepsilon/3$ and applying Lemma 4 on the graph G' this then allows him to efficiently estimate the distances between the *vertices* of G'. In particular, if the shortest path in G' between two vertices $u, v \in G'$ intersects Q then one can find an $(1+\varepsilon)$-approximation of their distance $d(u, v)$ in P using one of the constructed anchor points in $O(k)$ time.

However, we want an $(1+\varepsilon)$-approximation for the distance between any vertex in G and *any point* on Q; i.e. not just the vertices of Q and the intersection points of Q with edges of G. Thorup's construction does not immediately provide that, see for example Figure 5.

A continuous graph G^Q. We will instead define a slightly different graph G^Q that includes additional "Steiner" points on Q. This *does* allow us to accurately estimate the distance to *any* point on Q, and thus compute appropriate anchor points. We actually construct the cone graph G with parameter $\varepsilon_1 = \varepsilon/9$, and then extend it into a *continuous graph* $G^Q = (V^Q, E^Q)$ (see below). We obtain G^Q from G and Q as follows:

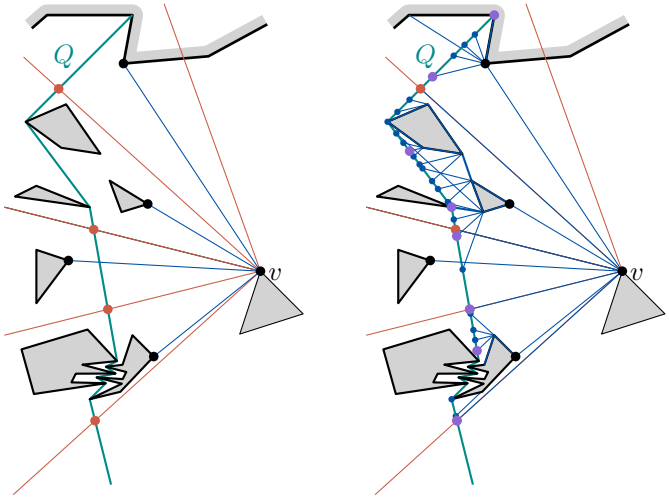


Figure 6 (a) The minimal cone neighbors $N(v)$ of vertex v, and the additional vertices $X_Q(v)$ (brown). (b) We compute a set of anchor points $A_Q(v)$ (purple) on Q based on G^Q (partially drawn).

- We include all vertices of G and of Q as vertices of G^Q.
- For each vertex $v \in G$ we consider the cones in the cone family centered at v. For each cone boundary ρ, we consider the first intersection point q of ρ with Q; if the resulting line segment $\overline{vq}$ lies inside P (i.e. the ray does not intersect the boundary of P before reaching q), we add q as a vertex, and (v, q) as an edge. Let $X_Q(v)$ denote the set of vertices added by v in this way. See Figure 6(a) for an illustration.
- We add all edges of G, and connect each pair of vertices that is consecutive along Q.

Observe that G^Q still has size $O(nk)$; as each vertex in G adds $O(k)$ additional vertices and edges, and Q also has at most $O(n)$ vertices. It is straightforward to construct G^Q from G in $O(nk \log n)$ time, using data structures for fixed-directional ray shooting queries [18].

▶ **Lemma 5.** *In $O(nk \log n)$ time we can construct a data structure of size $O(nk)$ so that for any point $s \in P$ we can compute the extra vertices $X_Q(s)$ in $O(k \log n)$ time. Hence, G^Q can be constructed in $O(nk \log n)$ time.*

Next, we argue that G^Q indeed allows us to ε_1-approximate the distances. We actually interpret G^Q as a *continuous graph* [5]. Every edge $(u, v) \in E^Q$ is a line segment $\overline{uv}$ in P, so we can extend the (graph) distance d_{G^Q} to points in the interior of edges. In particular, for points p, q on the same edge $\overline{uv}$ we define $d_{G^Q}(p, q) = \|p - q\|$ to simply be their Euclidean distance, and for $p \in \overline{uv} \in E^Q$ and $q \in \overline{wz} \in E^Q$ in the interior of different edges we have $d_{G^Q}(p, q) = \min_{u' \in \{u,v\}} \min_{w' \in \{w,z\}} \|p - u'\| + d_{G^Q}(u', w') + \|w' - q\|$. We are only interested in points in the interior of edges of Q. Hence, we prove:

▶ **Lemma 6.** *Let $u, v \in V$ be two polygon vertices in G^Q. Then we have that*

$$d(u, v) \leq d_{G^Q}(u, v) \leq (1 + \varepsilon_1) d(u, v).$$

Furthermore, for any point $q \in Q$ and any vertex $v \in G^Q$ we have that

$$d(v, q) \leq d_{G^Q}(v, q) \leq (1 + \varepsilon_1) d(v, q).$$

Computing and storing anchor points. We can now compute the distances in G^Q, and use them to construct an appropriate set of anchor points. See Figure 6(b). Due to Lemma 6 we can now actually just reuse Thorup's approach (i.e. Lemma 4) again:

▶ **Lemma 7.** *For every vertex v of P, we can compute a set $A_Q(v)$ of $O(k)$ anchor points on Q such that for any point q on Q we have $d(v,q) \leq d_{G^Q}(v, A_Q(v), q) \leq (1+\varepsilon)d(v,q)$.*

This takes $O(nk^2 \log n \log(kn))$ time in total.

3.3 The augmented cone graph

Let S be a dynamic set of point sites inside P. As we are interested in distances to (and between) the points in S, we also want to define and compute anchor points on Q for the points in S. The most straightforward approach would be to just build the above continuous graph G^Q on the set of points (vertices) $V \cup S$ rather than just the polygon vertices V. However, as computing the anchor points involves computing shortest paths in this graph, we cannot easily compute the anchor points for a single new site s. Similarly, maintaining (the distances in) the graph as S changes will be difficult. So, we argue that we can compute suitable anchor points for these new sites using a slightly different method instead.

Let $s \in P$ be a new point site. Similar to Section 3.1, we now consider the (continuous) graph $G^Q[s]$ that we obtain from G^Q by introducing $\{s\} \cup X_Q(s)$ as additional vertices, connecting s to every minimal cone neighbor in $N(s)$ and to every point in $X_Q(s)$, and subdividing the edges of Q to account for the additional vertices in $X_Q(s)$. This (continuous) graph still has $O(nk)$ vertices and $O(nk)$ edges. Moreover, we can use this graph to accurately estimate distances from s to any point on Q.

▶ **Lemma 8.** *Let s be a point in P. For any vertex w of P there is a path from s to w in $G^Q[s]$ of length $d(s,w) \leq d_{G^Q[s]}(s,w) \leq (1+\varepsilon_1)d(s,w)$ that passes through a cone neighbor $v \in N(s)$. Furthermore, for any point $q \in Q$ we have*

$$d(s,q) \leq d_{G^Q[s]}(s,q) \leq (1+\varepsilon_1)d(s,q).$$

Defining anchor points. Our goal is to define a set $A_Q(s)$ of anchor points on Q for point s. To this end, we first define a set of candidate points $X'_Q(s) = X_Q(s) \cup \bigcup_{v \in N(s)} A_Q(v)$ on Q. We can prove that if we assign appropriate weights to this set of $O(k^2)$ points it is indeed a set of anchor points. However, as we argue next, a subset of $O(k)$ of these points suffices.

Let $H = (V', E')$ be an undirected weighted graph with vertex set $V' = \{s\} \cup X'_Q(s)$. Vertex s is connected to each vertex $p \in X_Q(s)$ with an edge of length $\|s-p\| = d_{G^Q[s]}(s,p)$, and to $p \in A_Q(v)$ by an edge of length $\|s-v\| + d_{G^Q}(v,p)$. We connect vertices $u, w \in X'_Q(s)$ that appear consecutively on Q by an edge of length $d(u,w)$ (which is the length of the subpath along Q). Observe that the vertices of $X'_Q(s)$ form a shortest path in the graph H.

▶ **Lemma 9.** *Let $\varepsilon_2 > 0$ be a parameter such that $O(1/\varepsilon_2) = O(k)$. In $O(k^2)$ time we can compute a set $A_Q(s)$ of $O(k)$ discrete anchor points for H, i.e. so that for any $p \in X'_Q(s)$*

$$d_H(s,p) \leq d_H(s, A_Q(s), p) \leq (1+\varepsilon_2)d_H(s,p).$$

We use Lemma 9 with $\varepsilon_2 := \varepsilon/9$ to compute a set $A_Q(s)$ of $O(k)$ anchor points for s with respect to H. Hence, for each point $p \in X'_Q(s)$ (so in particular for those in a set $A_Q(v)$) there is a path of length $(1+\varepsilon_2)d_H(s,p) \leq (1+\varepsilon_2)(\|s-v\| + d_{G^Q}(v,p))$ via an anchor point in $A_Q(s)$. Next, we argue that this set is a valid set of anchor points for any point on Q.

▶ **Lemma 10.** *Let s be a point in P. In $O(k \log n + k^2)$ time, we can compute a set $A_Q(s)$ of $O(k)$ anchor points, so that for any point $q \in Q$ we have*

$$d(s,q) \leq d_{G^Q[s]}(s, A_Q(s), q) \leq (1+\varepsilon)d(s,q).$$

Estimating distances via the shortest path Q. Given a shortest path Q, and points $s, t \in P$ whose shortest path $\pi(s,t)$ in P intersects Q, we can use the above tools to accurately estimate the distance between s and t. More specifically, we compute G^Q and the anchor points as defined above. Additionally, we fix an endpoint u of Q and compute and store, for each vertex w of Q the distance to u (i.e. the path length along Q). This can easily be done in $O(n)$ additional time. Given a pair of points a, b on Q (and the edges that contain a and b, respectively) we can then compute their distance $d(a,b)$ (i.e. the length of the subpath $Q[a,b]$) in constant time.

Let $a \in A_Q(s)$ and $b \in A_Q(t)$ be anchor points of s and t on Q, respectively. The subpath $\hat{\pi}_a$ from s to a in $G^Q[s]$ corresponds to an obstacle avoiding path in P. Similarly, the subpath $\hat{\pi}_b$ from b to t is obstacle avoiding. Furthermore, since Q is a shortest (obstacle-avoiding) path in P, so is its subpath $Q[a,b]$ from a to b. So, the concatenation of $\hat{\pi}_a$, $Q[a,b]$, and $\hat{\pi}_b$ connects s and t and is obstacle avoiding, and has length $\hat{d}_a + \hat{d}_b + d(a,b)$. Now define

$$\hat{d}_Q(s,t) = \min_{a \in A_Q(s), b \in A_Q(t)} \hat{d}_a + \hat{d}_b + d(a,b)$$

to be the length of a shortest such obstacle avoiding path $\hat{\pi}_Q(s,t)$. We then have:

▶ **Lemma 11.** *Let P be a polygonal domain with n vertices, and let Q be a shortest path in P. In $O(nk^2 \log n \log(kn))$ time, we can construct a data structure on P and Q of size $O(nk)$ that can answer the following queries in $O(k \log n + k^2)$ time: given any pair of points $s, t \in P$ for which a shortest path $\pi(s,t)$ intersects Q, return the length $\hat{d}_Q(s,t)$ of an obstacle avoiding path $\hat{\pi}_Q(s,t)$ so that $d(s,t) \leq \hat{d}_Q(s,t) \leq (1+\varepsilon)d(s,t)$.*

4 A data structure for answering distance queries

In this section we develop an $O(nk \log n)$ space data structure that stores the polygonal domain P, so that given a pair of query points s, t we can compute an $(1+\varepsilon)$-approximation of the distance $d(s,t)$ in $O(k^2 \log n)$ time. Our data structure is essentially a balanced hierarchical subdivision of the polygonal domain combined with the graph-based approach to estimate distances from the previous section.

A balanced hierarchical subdivision for polygonal domains. A balanced hierarchical subdivision [9] is a recursive subdivision of a simple polygon into subpolygons. It is a crucial ingredient to obtain efficient solutions to many problems involving simple polygons [1, 13]. As was also recently observed [12], Thorup's approach [19] essentially provides a balanced hierarchical subdivision for polygonal domains. The following lemma summarizes this result:

▶ **Lemma 12** ([19, 12])**.** *Let P be a polygonal domain with n vertices. In $O(n \log n)$ time, one can compute a tree $\mathcal{T}$ so that*

1. *the leaves of $\mathcal{T}$ correspond to triangles in a triangulation of P,*
2. *each node ν of T corresponds to a subpolygon P_ν,*
3. *each internal node ν also corresponds to a set $\mathcal{Q}_\nu$ of shortest paths so that:*
   a. *$\mathcal{Q}_\nu$ consists of at most three shortest paths (see Figure 7),*
   b. *every shortest path $Q \in \mathcal{Q}_\nu$ is also a shortest path in P_ν, and*
   c. *the paths in $\mathcal{Q}_\nu$ partition P_ν into P_μ and P_ω, corresponding to the children of ν,*

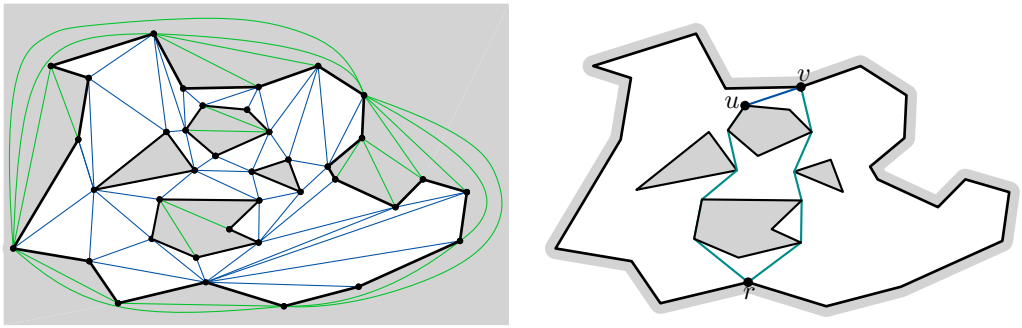


Figure 7 One step in the construction of the balanced hierarchical separator.

4. *the height of the tree is* $O(\log n)$*, and*
5. *the total complexity of the subpolygons* P_ν *on each level of* $\mathcal{T}$ *is* $O(n)$*, and thus the total complexity of* $\mathcal{T}$ *is* $O(n \log n)$*.*

The data structure. Our data structure now stores the balanced hierarchical subdivision $\mathcal{T}$ from Lemma 12. We build a point location data structure [18] on the triangulation induced by the leaves of $\mathcal{T}$. Additionally, for each edge e of this triangulation, let ν be the highest node in $\mathcal{T}$ for which the edge appears on a shortest path $Q \in \mathcal{Q}_\nu$ (if such a node exists). We store a pointer from edge e to this shortest path. We can easily compute these pointers during the construction of $\mathcal{T}$.

Consider a node ν. For each of the $O(1)$ shortest paths $Q \in \mathcal{Q}_\nu$ we will construct the data structure from Lemma 11 (i.e. the graph G^Q and the sets of anchor points) with respect to the subpolygon P_ν. Since the total size of all subpolygons P_ν on each level of the tree is $O(n)$, the total size of all of these data structures on a given level of $\mathcal{T}$ is $O(kn)$, and thus $O(nk \log n)$ in total. Similarly, the total preprocessing time is $O(nk^2 \log^2 n \log(kn))$.

Query. Given a pair of query points s, t we find the leaf triangles Δ_s and Δ_t that contain points s and t, respectively. If s and t lie in the same triangle, we can trivially report the length $\overline{st}$ as their shortest path. Otherwise, a shortest path from s to t must intersect one of the shortest paths $Q \in \mathcal{Q}_\nu$ for a common ancestor ν of the two leaves. In particular, let μ be the highest node of $\mathcal{T}$ for which $\pi(s,t)$ intersects a shortest path $Q \in \mathcal{Q}_\mu$. It follows that the shortest path $\pi(s,t)$ lies inside P_μ, and thus we can use the Lemma 11 data structure for Q to obtain a path of length at most $(1+\varepsilon)d(s,t)$. Unfortunately, we cannot easily find this node μ, so we query all Lemma 11 data structures for all shortest paths $Q \in \mathcal{Q}_\nu$, for all common ancestors ν of the leaves representing Δ_s and Δ_t, and report the shortest distance found. If either s or t lies on an edge e of the boundary of some triangle that is part of some shortest path $Q \in Q_\nu$, we use the pointer stored at e to jump to the highest such node, and query only the common ancestors of these nodes instead. Since every query returns the length of an obstacle-avoiding path, the returned estimate $\hat{d}(s,t)$ satisfies $d(s,t) \leq \hat{d}(s,t) \leq (1+\varepsilon)d(s,t)$ as desired. Since each query takes $O(k \log n + k^2)$ time, the total query time is $O(k \log^2 n + k^2 \log n)$.

Improving the query time. The $O(k \log^2 n)$ term in the query time is only due to computing the ray shooting queries needed to compute the sets $X_Q(s)$ for each shortest path Q considered. Next, we argue that we can compute these sets in only $O(k \log n)$ time by building an additional $O(nk \log n)$ space data structure. The overall space usage thus remains the same, and the final query time becomes $O(k^2 \log n)$ as claimed.

▶ **Lemma 13.** *In $O(nk \log n)$ time, we can build an $O(nk \log n)$ space data structure that given a query point s, allows us to compute the sets $X_Q(s)$ for all shortest paths $Q \in \bigcup_{\nu \in \mathcal{T}(s)} \mathcal{Q}_\nu$, where $\mathcal{T}(s)$ is the path in $\mathcal{T}$ from the root to the leaf triangle containing s.*

Finally, we plug in that $k = \lceil \frac{1}{\varepsilon} \rceil$, which establishes Theorem 2.

▶ Remark 14. Thorup's final approach [19] uses one additional idea: that one can guarantee that the boundary of a subgraph G_ν (used during the construction in Lemma 12) intersects only $O(1)$ shortest paths from ancestor separator paths. Therefore, he can also store anchor points on these paths, and query only the shortest paths of the lowest common ancestor of the leaves corresponding to s and t. It is not clear to us if this idea is still applicable with our approach. For a shortest path B on the boundary of some subpolyon P_ν, it is not clear in which (sub)polygon to place the path B so that we can invoke Lemma 11.

5 Dynamic nearest neighbor searching

In this section, we develop our data structure for approximate nearest neighbor searching with a dynamic set of sites S in a polygonal domain P. In Section 5.1 we first develop a simple solution for when S is simply a set of sites in the plane. In Section 5.2 we consider how to solve a restricted case in which we are given a shortest path Q in P, and we have to only answer queries q for which the shortest path between q and any site in S intersects Q. Then finally, in Section 5.3 we combine these results with the balanced hierarchical subdivision to obtain our data structure for the general problem.

5.1 Dynamic Euclidean ε-close neighbor searching

Let S be a dynamic set of m points in $\mathbb{R}^2$. We present a simple $O(km \log m)$ space data structure that given a query point q can report a site s for which the distance $d(q, s)$ is at most $(1+\varepsilon)d(q, s^*)$ in $O(k \log m)$ time. Here, s^* is a site in S closest to q. Our data structure supports insertions and deletions in $O(k \log m)$ time as well. Our data structure essentially uses the cone based approach from Section 3.1; i.e. for each cone C in the cone family $\mathcal{F}_\varepsilon$, we build a dynamic data structure that can report the site $s \in S \cap C_q$ with minimum cone distance. By the argument from Lemma 3 it follows that this then yields an ε-close site.

▶ **Lemma 15.** *Let S be a set of m sites in $\mathbb{R}^2$. There is an $O(km \log m)$ space data structure that, given a query point $q \in \mathbb{R}^2$ can report an $1/k$-close site from S in $O(k \log^2 m)$ time. Insertions and deletions of sites are supported in amortized $O(k \log^2 m)$ time.*

5.2 Maintaining an additively weighted Voronoi diagram

Let S be the set of m sites in the polygonal domain P, and let Q be a shortest path. We describe a dynamic data structure that, given a query point $q \in P$ can report an ε-close site $s \in S$, provided that the shortest path from q to the closest site $s^* \in S$ intersects Q. More specifically, our goal is to compute a site s which achieves the following minimum

$$\min_{s \in S} \hat{d}_Q(q, s) = \min_{s \in S} \min_{a \in A_Q(q), b \in A_Q(s)} \hat{d}_a + \hat{d}_b + d(a, b). \tag{1}$$

If the shortest path between s^* and q intersects Q, then by Lemma 11 this value is indeed at most $(1+\varepsilon)d(s^*, q)$, and thus the site s that achieves this minimum is ε-close to q. We rewrite Equation 1 to $\min_{a \in A_Q(q)} L_Q(a)$, where we define $L_Q(a) := \hat{d}_a + \min_{s \in S, b \in A_Q(s)} \hat{d}_b + d(a, b)$. Finding the optimal b for the evaluation of $L_Q(a)$ is essentially the closest site problem

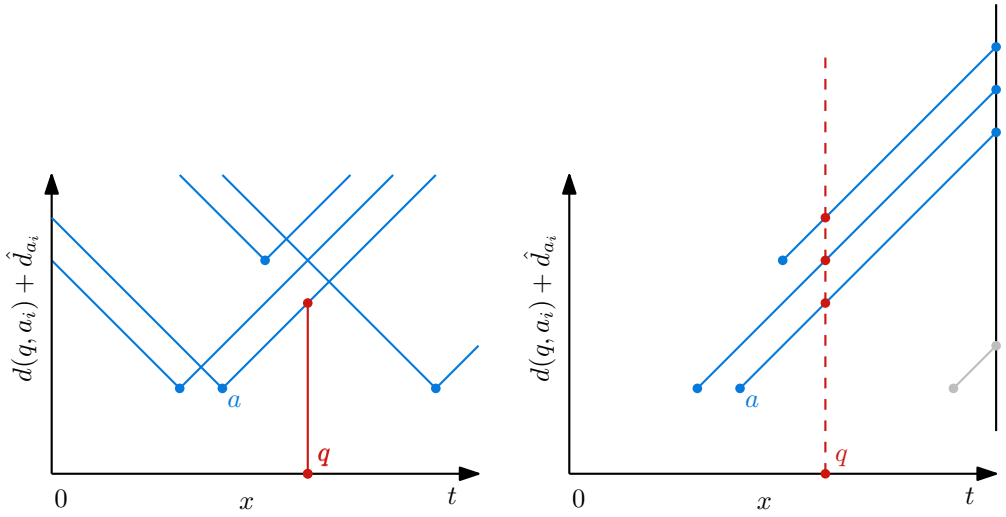


Figure 8 (a) For a query q we want to find the lowest intersection of the line at $q = x$ with the graph of the function $d(a, \cdot)$ of an anchor point $a \in A_Q(s)$. (b) The right half-lines of our V-functions. Note that the half lines intersect the vertical line at q in the same order as at t. Our data structure would find the site to the left of $q = x$ which makes the lowest intersection with $x = t$.

with additive weights $\hat{d}_b$. Furthermore, we only have to consider a 1-dimensional closest site problem as all anchor points lie on a shortest path $Q = \pi(u, v)$. This means we can map the coordinates of each point $a \in Q$ to the one-dimensional coordinate $a_x = d(u, a)$, this preserves distances as for all $a, b \in \pi(u, v)$ we have $d(a, b) = |d(u, b) - d(u, a)| = d(a_x, b_x)$.

Since these functions a have nice structure (see Figure 8), we can relatively easily maintain and query L_Q using priority search trees [4]. We store all $O(mk)$ anchor points (constructed using Lemma 11) and obtain:

▶ **Lemma 16.** *Let P be a polygonal domain with n vertices, let Q be a shortest path in P, and let S be a set of m sites in P. There is an $O(nk + mk)$ space data structure such that:*

- *For any point $q \in P$ we can find $\min_{s \in S} \hat{d}_Q(q, s)$ and the site which achieves this minimum in $O(k \log n + k^2 + k \log(mk))$ time.*
- *We can insert sites in amortized $O(k \log n + k^2 + k \log(mk))$ time, and delete sites in amortized $O(k \log(mk))$ time.*

Constructing an initially empty data structure takes $O(nk^2 \log n \log(kn))$ time.

5.3 The final data structure

Our data structure for general approximate nearest neighbor queries uses the same approach as in Section 4. Our data structure consists of the balanced hierarchical separator $\mathcal{T}$ from Lemma 12, in which we store associated structures at each node. We again preprocess the triangulation formed by the leaves for efficient point location, and store a pointer from each edge e to the highest shortest path Q in some $\mathcal{Q}_\nu$ that uses edge e.

For each internal node ν, and for each shortest path $Q \in \mathcal{Q}_\nu$, we now build an associated Lemma 16 data structure on Q. This structure will store a subset $S_{\nu,Q}$ of the sites. In particular, the sites in $S \cap P_\nu$ that did not lie on shortest paths of ancestors of ν.

For each leaf ν of $\mathcal{T}$, which represents some triangle P_ν, we store (a subset S_ν of) the sites in $S \cap P_\nu$ in the data structure from Lemma 15. We again store only the sites that did not already appear on shortest paths stored at ancestors of ν.

Finally, we build the data structure from Lemma 13 that allows us to efficiently compute the sets $X_Q(s)$ on any root-to-leaf path.

Space and construction time. Since the total complexity of all polygons associated with $\mathcal{T}$ is $O(n \log n)$, it readily follows that the total space of all Lemma 16 data structures is $O(nk \log n + mk \log n)$. As each site also appears in at most one leaf, it then follows that we use a total of $O(nk \log n + mk \log n + mk \log m)$ space. Similarly, constructing an empty data structure takes $O(nk^2 \log^2 n \log(kn))$ time.

Answering a query. Our goal is to report a site $s \in S$ that is ε-close to a query $q \in P$.

We point locate to find a triangle Δ_q that contains q, and compute the sets $X_Q(s)$ for all nodes on the path $\mathcal{T}(q)$ towards this leaf using the Lemma 13 data structure. For each shortest path $Q \in \mathcal{Q}_\nu$ of each node $\nu \in \mathcal{T}(q)$ along this path, we then query the Lemma 16 data structure. Each such query returns a site $s_{\nu,Q}$ that minimizes $\hat{d}_Q(q, s_{\nu,Q})$ among the sites in $S_{\nu,Q}$ (as well as the actual distance estimate $\hat{d}_Q(q, s_{\nu,Q})$). Finally, we query the Lemma 15 data structure of the leaf triangle Δ_q, to compute an ε-close site from the subset of sites stored there. We report the site s with overall minimum distance estimate.

Let s^* be site in S closest to q. We now argue that the site s and the distance estimate $\hat{d}(s, q)$ that we return indeed satisfies $d(q, s^*) \leq d(q, s) \leq \hat{d}(s, q) \leq (1 + \varepsilon)d(q, s^*)$.

If s^* also lies in Δ_q, then Lemma 15 guarantees that s is within distance $(1 + \varepsilon)d(q, s^*)$. Otherwise, the shortest path $\pi(q, s^*)$ from q to s^* must intersect one of the shortest paths $Q \in \mathcal{Q}_\nu$ on one of the nodes on the root-to-leaf path $\mathcal{T}(q)$. In particular, consider a shortest path of the highest such node ν. It follows that s^* is one of the sites in $S_{\nu,Q}$, and thus Lemma 16 guarantees that the candidate $s_{\nu,Q}$ has the minimum $\hat{d}_Q(q, s_{\nu,Q})$ value among all sites in $S_{\nu,Q}$. Hence, our distance estimate satisfies $\hat{d}(q, s) \leq \hat{d}_Q(q, s_{\nu,Q}) \leq \hat{d}_Q(q, s^*)$. Lemma 11 tells us that $\hat{d}_Q(q, s^*) \leq (1 + \varepsilon)d(q, s^*)$ as desired. Finally, since our distance estimate $\hat{d}(q, s)$ corresponds to the length of an obstacle avoiding path, we also have $d(s, q^*) \leq d(s, q) \leq \hat{d}(q, s)$, and thus s is indeed ε-close to q.

Finding the triangle containing q, and traversing the root-to-leaf path corresponding to this triangle takes $O(\log n)$ time. Querying all $O(\log n)$ Lemma 16 data structures along this path would take a total of $O(k \log^2 n + k^2 \log n + k \log(mk) \log n)$ time. Finally, the query to the Lemma 15 data structure stored at the leaf takes $O(k \log^2 m)$ time. As before, the $O(k \log^2 n)$ term in the query time is only due to computing the sets $X_Q(s)$ in every node separately. By querying the Lemma 13 data structure, we can again reduce this term to $O(k \log n)$. This reduces the total time required to query the Lemma 16 data structures to $O(k^2 \log n + k \log(mk) \log n + k \log^2 m)$, which simplifies to $O(k^2 \log n + k \log n \log m + k \log^2 m)$. As this term still dominates the other terms, this is also the final query time.

Updates. We first describe the process for inserting a new site s into S. We point locate to find a triangle P_ν that contains s. When s lies in the interior of such a triangle, then it must lie in the interior of all ancestor subpolygons P_μ as well, so we insert s into the Lemma 15 data structure of the leaf ν, and the Lemma 16 data structures along the root-to-leaf path. If s lies on an edge e of P_ν that is also an edge in a shortest path in one of the separators, we use the pointer stored at this edge to jump to the shortest path $Q \in \mathcal{Q}_{\nu'}$ of the highest such node ν', and insert s in the Lemma 16 data structures of ν' and its ancestors.

We have to insert s into at most one Lemma 15 data structure, (in $O(k \log^2 m)$ time), and at most $O(\log n)$ Lemma 16 data structures. Naively inserting them into every such structure then gives a total insertion time of $O(k \log^2 n + k^2 \log n + k \log(mk) \log n + k \log^2 m)$. As with the queries, we can compute the sets $X_Q(s)$ in $O(k \log n)$ time in total using Lemma 13. This then gives us a total insertion time of $O(k^2 \log n + k \log n \log m + k \log^2 m)$.

When we delete a site, we simply delete the site from all places where it was inserted. This takes a total of $O(k \log n \log m + k \log^2 m)$ time. Plugging in $k = \lceil 1/\varepsilon \rceil$ gives us Theorem 1.

References

1 Pankaj K. Agarwal, Lars Arge, and Frank Staals. Improved dynamic geodesic nearest neighbor searching in a simple polygon. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, volume 99 of *LIPIcs*, pages 4:1–4:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.SOCG.2018.4.

2 Pankaj K. Agarwal and Jiří Matoušek. Dynamic Half-Space Range Reporting and its Applications. *Algorithmica*, 13(4):325–345, 1995. doi:10.1007/BF01293483.

3 Jon Louis Bentley and James B Saxe. Decomposable searching problems I. Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980. doi:10.1016/0196-6774(80)90015-2.

4 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.

5 Sergio Cabello, Delia Garijo, Antonia Kalb, Fabian Klute, Irene Parada, and Rodrigo I. Silveira. Algorithms for distance problems in continuous graphs. In Pat Morin and Eunjin Oh, editors, *19th International Symposium on Algorithms and Data Structures, WADS 2025, August 11-15, 2025, York University, Toronto, Canada*, volume 349 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.WADS.2025.13.

6 Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010. doi:10.1145/1706591.1706596.

7 Timothy M. Chan. Dynamic generalized closest pair: Revisiting eppstein's technique. In Martin Farach-Colton and Inge Li Gørtz, editors, *3rd Symposium on Simplicity in Algorithms, SOSA 2020, Salt Lake City, UT, USA, January 6-7, 2020*, pages 33–37. SIAM, 2020. doi:10.1137/1.9781611976014.6.

8 Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. *Discret. Comput. Geom.*, 64(4):1235–1252, 2020. doi:10.1007/S00454-020-00229-5.

9 Bernard Chazelle and Leonidas J. Guibas. Visibility and intersection problems in plane geometry. *Discrete & Computational Geometry*, 4:551–581, 1989. doi:10.1007/BF02187747.

10 K. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 56–65, New York, NY, USA, 1987. Association for Computing Machinery. doi:10.1145/28395.28402.

11 Sarita de Berg, Tillmann Miltzow, and Frank Staals. Towards space efficient two-point shortest path queries in a polygonal domain. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece*, volume 293 of *LIPIcs*, pages 17:1–17:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SOCG.2024.17.

12 Sarita de Berg, Frank Staals, and Marc J. van Kreveld. The complexity of geodesic spanners. *J. Comput. Geom.*, 15(1):21–65, 2024. doi:10.20382/JOCG.V15I1A2.

13 Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *J. Comput. Syst. Sci.*, 39(2):126–152, 1989. doi:10.1016/0022-0000(89)90041-X.

14 John Hershberger and Subhash Suri. An optimal algorithm for euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999. doi:10.1137/S0097539795289604.

15 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar voronoi diagrams for general distance functions and their algorithmic applications. *Discret. Comput. Geom.*, 64(3):838–904, 2020. doi:10.1007/S00454-020-00243-7.

16 Chih-Hung Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. *SIAM J. Comput.*, 51(3):723–765, 2022. doi:10.1137/20M1388371.

17 Eunjin Oh and Hee-Kap Ahn. Voronoi diagrams for a moderate-sized point-set in a simple polygon. *Discret. Comput. Geom.*, 63(2):418–454, 2020. doi:10.1007/S00454-019-00063-4.

18 Neil Sarnak and Robert Endre Tarjan. Planar point location using persistent search trees. *Commun. ACM*, 29(7):669–679, 1986. doi:10.1145/6138.6151.

19 Mikkel Thorup. Compact oracles for approximate distances around obstacles in the plane. In Lars Arge, Michael Hoffmann, and Emo Welzl, editors, *Algorithms – ESA 2007*, pages 383–394, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi:10.1007/978-3-540-75520-3_35.

20 Joost van der Laan, Frank Staals, and Lorenzo Theunissen. Approximate dynamic nearest neighbor searching in a polygonal domain, 2026. arXiv:2603.11775.

21 Haitao Wang. A new algorithm for euclidean shortest paths in the plane. *J. ACM*, 70(2):11:1–11:62, 2023. doi:10.1145/3580475.

Tree-Like Shortcuttings of Trees

Hung Le
University of Massachusetts Amherst, MA, USA

Lazar Milenković
Tel Aviv University, Israel
INSAIT, Sofia University "St. Kliment Ohridski", Bulgaria

Shay Solomon
Tel Aviv University, Israel

Cuong Than
University of Massachusetts Amherst, MA, USA

Abstract

Sparse *shortcuttings* of trees – equivalently, sparse 1-spanners for tree metrics with bounded *hop-diameter* – have been studied extensively (under different names and settings), since the pioneering works of [24, 14, 2, 7], initially motivated by applications to range queries, online tree product, and MST verification, to name a few. These constructions were also lifted from trees to other graph families using known low-distortion embedding results. The works of [24, 14, 2, 7] establish a tight tradeoff between *hop-diameter* and sparsity (or average degree) for tree shortcuttings and imply constant-hop shortcuttings for n-node trees with sparsity $O(\log^* n)$. Despite their small sparsity, **all known constant-hop shortcuttings contain dense subgraphs** (of sparsity $\Omega(\log n)$), which is a significant drawback for many applications.

We initiate a systematic study of constant-hop tree shortcuttings that are "tree-like". We focus on two well-studied graph parameters that measure how far a graph is from a tree: *arboricity* and *treewidth*. Our contribution is twofold.

- **New upper and lower bounds for tree-like shortcuttings of trees**, including an *optimal* tradeoff between hop-diameter and treewidth for all hop-diameter up to $O(\log \log n)$. We also provide a lower bound for larger values of k, which together yield hop-diameter $\times$ treewidth $= \Omega((\log \log n)^2)$ *for all values of hop-diameter*, resolving an open question of [16, 19].
- **Applications of these bounds**, focusing on low-dimensional Euclidean and doubling metrics. A seminal work of Arya et al. [4] presented a $(1+\epsilon)$-spanner with constant hop-diameter and sparsity $O(\log^* n)$, but with large arboricity. We show that constant hop-diameter is sufficient to achieve arboricity $O(\log^* n)$. Furthermore, we present a $(1+\epsilon)$-stretch routing scheme in the *fixed-port* model with 3 hops and a local memory of $O(\log^2 n / \log \log n)$ bits, resolving an open question of [18].

2012 ACM Subject Classification Theory of computation → Sparsification and spanners

Keywords and phrases spanner, tree shortcutting, arboricity, treewidth

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.70

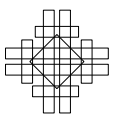
Related Version *Full Version*: `https://arxiv.org/abs/2510.14918`

Funding *Hung Le*: Supported by NSF grant CCF-2517033 and NSF CAREER Award CCF-2237288.
Lazar Milenković: Funded by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, and the United States National Science Foundation (NSF). This research was partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).
Shay Solomon: Funded by the European Union (ERC, DynOpt, 101043159). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority



© Hung Le, Lazar Milenković, Shay Solomon, and Cuong Than;
licensed under Creative Commons License CC-BY 4.0





can be held responsible for them. Also funded by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, and the United States National Science Foundation (NSF).
Cuong Than: Supported by NSF grant CCF-2517033 and NSF CAREER Award CCF-2237288. Also supported by a Google Ph.D. Fellowship.

1 Introduction

Given a tree $T = (V, E)$ and an integer $k \geq 1$, a *tree shortcutting* of T with hop-diameter k is a graph $G = (V, E')$ such that for every two vertices $u, v \in V$, there is a path in G consisting of at most k edges such that $\delta_G(u, v) = \delta_T(u, v)$, where $\delta_G(u, v)$ represents the distance between u and v in G. The problem of constructing sparse tree shortcuttings with small hop-diameter has been studied extensively (under different names and settings) since the pioneering works of [24, 14, 2, 7]. The optimal tradeoff is hop-diameter k with $\Theta(n\alpha_k(n))$ edges, where $\alpha_k(n)$ is a very slowly-growing inverse Ackermann function, $\alpha_2(n) = \lceil \log n \rceil$, $\alpha_3(n) = \lceil \log \log n \rceil$, $\alpha_4(n) = \log^* n$, etc.[1] (See the full version of the paper for a formal definition.)

Tree shortcutting has many applications, for example, finding max-flow [23], MST verification [23], maintaining MST under edge weight increases [23], computing semigroup product along paths and trees [24, 2], to name a few. In these applications, the shortcut graph serves as a compact data structure where k determines the query time, and the number of edges is the space. Tree shortcutting is a key subroutine in constructing low-hop $(1 + \epsilon)$-spanners for Euclidean [4, 22, 21] and doubling metrics [9, 18]. (See Section 2 for a formal definition.)

The fundamental drawback of sparse tree shortcuttings is that they are not uniformly-sparse – all of the aforementioned tree shortcutting constructions contain subgraphs with average degree of $\Omega(\log n)$ for $k = 2$ and $\Omega(\sqrt{n})$ for $k = 3$. As k increases, the (global) sparsity of the shortcutting reduces substantially, but the uniform sparsity remains $\Omega(\log n)$. Motivated by this, we initiate a systematic study of low-hop tree shortcuttings that are "tree-like". In particular, we are interested in two notions that capture uniform sparsity: *treewidth* and *arboricity*. (See Section 2 for formal definitions.)

Low-treewidth shortcutting. Low-treewidth shortcutting was introduced in the context of low-treewidth embedding of planar metrics. Specifically, Filtser and Le [16] showed that if one can construct a tree shortcutting with hop-diameter k and treewidth t, then one obtains an embedding of planar metrics into graphs of treewidth $O(kt/\epsilon)$ and additive distortion $+\epsilon D$, where D is the diameter of the input metric. They constructed a tree shortcutting with hop-diameter $k = O(\log \log n)$ and treewidth $t = O(\log \log n)$, and hence obtained an embedding with treewidth $O((\log \log n)^2/\epsilon)$. This bound does not seem to be optimal – other works [17, 11] used different embedding techniques to remove the dependency on n but at the cost of a (much) higher dependency on $1/\epsilon$. If one can construct a shortcutting with hop-diameter $k = O(1)$ and treewidth $t = O(\alpha_k(n))$ (which matches the sparsity bound), then one gets an embedding of planar metrics with treewidth $O(\alpha_k(n)/\epsilon) = O(\log^*(n)/\epsilon)$, which is almost as good as $O(1/\epsilon)$. The linear dependency on $1/\epsilon$ is optimal [16], whereas the other embedding techniques [17, 11] have an inherent $\Omega(1/\epsilon^2)$ barrier. Motivated by this observation, Filtser and Le [16] asked:

[1] Throughout the paper, we use $\log(x) \coloneqq \log_2(x)$.

▶ **Question 1** ([19, 16]). *Is there a tree shortcutting with treewidth t and hop-diameter k such that $k \cdot t = o((\log\log n)^2)$? Furthermore, is it possible for $k \cdot t$ to approach $O(1)$?*

In this work, we seek to answer a **broader question**: *What is the exact tradeoff between the hop-diameter k and the treewidth t of tree shortcuttings?* Understanding the full tradeoff would not only answer Question 1 but could also be useful for other applications that require a different hop-diameter. Specifically, in this paper, we use our construction for $k = 3$ to resolve an open problem posed by Kahalon et al. [18] regarding compact routing schemes.

Low-arboricity shortcutting. Given a set of points P in an Euclidean (or doubling) space of dimension d, one can construct a sparse $(1+\epsilon)$-spanner with a small hop-diameter. This is achieved via tree covers. (See Section 2 for a definition.) Known tree cover constructions for Euclidean and doubling metrics achieve stretch $1+\epsilon$ using $\epsilon^{-O(d)}$ trees [4, 5, 12]. By constructing a k-hop shortcutting G_T for each tree T in the cover $\mathcal{T}$, the union $\cup_{T\in\mathcal{T}} G_T$ is a k-hop $(1+\epsilon)$-spanner. For a constant ϵ and d, the sparsity is only $O(1)$ times worse than the sparsity of the shortcuttings, which is $O(\alpha_k(n))$. However, if G_T has low treewidth for every tree T, the union $\cup_{T\in\mathcal{T}} G_T$ might have a very large treewidth. A recent work on low-treewidth Euclidean spanners by Buchin, Rehs, and Scheele [8] showed that treewidth-t spanners must have stretch $\Omega(n/t^{d/(d-1)})$. That is, if the stretch is $O(1)$, the treewidth has to be very large: $\Omega(n^{1-1/d})$.

While the treewidth grows substantially under taking union, the arboricity does not: if G_T has arboricity at most β for every $T \in \mathcal{T}$, then $\cup_{T\in\mathcal{T}} G_T$ has arboricity $|\mathcal{T}|\beta$, which is $O(\beta)$ for constant ϵ and d. This makes arboricity an appealing tree-like measure. Specifically, one could hope to construct a low-arboricity Euclidean (or doubling) $(1+\epsilon)$-spanner with small hop-diameter using low-arboricity tree shortcuttings.

1.1 Our contribution

Our key **conceptual contribution** is in identifying the power of tree-like shortcuttings of trees, in particular in the regime of very small hop-diameter. We employ the low-treewidth construction with hop-diameter 3 to improve the state-of-the-art on low-hop compact routing schemes in tree and doubling metrics. Using the low-arboricity shortcuttings, we devise a $(1+\epsilon)$-spanner with arboricity $O(\log^* n)$ and hop-diameter 6 for doubling metrics. We also fully resolve Question 1 in the negative, which is our **main technical contribution**.

Treewidth. Our main technical contribution is a lower bound on the treewidth t of tree shortcuttings for any hop-diameter k, which holds even when the underlying graph is a path. This result fully resolves Question 1 in the negative.

▶ **Theorem 2.** *For every $n \geq 1$, every shortcutting with hop-diameter k for an n-point path must have treewidth:*

- $t = \Omega(k \log^{2/k} n)$ *for even k and* $t = \Omega(k(\frac{\log n}{\log\log n})^{2/(k-1)})$ *for odd $k \geq 3$, whenever* $k \leq \frac{2}{\ln(2e)} \ln\ln n$;
- $t = \Omega((\log\log n)^2/k)$ *whenever* $k > \frac{2}{\ln(2e)} \ln\ln n$.

Specifically, Theorem 2 implies that $t \cdot k = \Omega((\log\log n)^2)$, and therefore, the artificially looking upper bound by Filtser and Le [16] is indeed optimal.

We prove Theorem 2 by constructing a large clique minor K_t in any given shortcutting H with hop-diameter k. Such a clique minor is a certificate that the treewidth of G is at least t. Our lower bound instance is an n-vertex path, and the proof is by induction on

k. We observe that when $k = 2$, shortcutting H contains the clique $K_{\lfloor \log n \rfloor + 1}$ as a minor. For a higher even value of $k \geq 4$, we use the induction hypothesis for $k - 2$. We divide the path into equally sized subpaths and consider two cases. If there is a subpath where every vertex has an edge going out of it, we can readily construct a large minor using this subpath. Otherwise, each subpath P has at least one vertex without an edge going out of P. This allows us to use the induction hypothesis for $k - 2$ on a graph obtained by contracting each subpath into a single vertex. The argument for odd values of k is analogous. Although the method of constructing the minor is simple to describe, the analysis is much more intricate, specifically handling the dependency between k and n in the recurrences. See Section 3 for details.

We also give an upper bound construction that matches the lower bound when $k = O(\log \log n)$, which arguably is the most interesting regime. Our construction is a rather natural adaptation of the known result by Filtser and Le [16]. The difference is that when we shortcut a set of "cut vertices" recursively, we set the hop bound to be $k - 2$. This requires some technical work; the proofs are deferred to the full version. The following theorem summarizes the construction.

▶ **Theorem 3.** *For every $n \geq 1$ and every $k = O(\log \log n)$, every n-vertex tree admits a shortcutting with hop-diameter k and treewidth $O(k \log^{2/k} n)$ for even k and $O(k(\frac{\log n}{\log \log n})^{2/(k-1)})$ for odd $k \geq 3$.*

In Section 5, we use this construction to devise a 3-hop routing scheme for tree metrics with stretch 1 and $O(\log^2 n / \log \log n)$ bits per vertex. This answers the open question from [18]: "Whether or not one can use a spanner of larger (sublogarithmic and preferably constant) hop-diameter for designing compact routing schemes with $o(\log^2 n)$ bits is left here as an intriguing open question." In the full version of the paper, we prove that the bound is tight. Using tree covers, we obtain a 3-hop routing scheme with stretch $(1 + \epsilon)$ that uses $\epsilon^{-\tilde{O}(d)} \log^2 n / \log \log n$ bits per vertex for every n-point metric with doubling dimension d. (See Section 5.)

▶ **Theorem 4.** *For every n and every n-point metric with doubling dimension d, there is a 3-hop routing scheme with stretch $(1 + \epsilon)$ that uses $\epsilon^{-\tilde{O}(d)} \cdot \log^2 n / \log \log n$ bits per vertex.*

This provides the first routing scheme in Euclidean and doubling metrics, where the number of hops is $o(\log n)$, and the labels consist of $o(\log^2 n)$ bits.

Arboricity. In Section 4 we focus on low-arboricity shortcuttings. We show how to construct shortcuttings of *hierarchically separated trees (HSTs)* where the arboricity grows proportionally to an inverse Ackermann function.

Our starting point is a sparse shortcutting of an n-vertex path. We use the well-known fact about arboricity – if the edges of a given graph can be oriented so that the in-degree is bounded by β, then the arboricity is at most $\beta + 1$. Our key observation is that one can split every edge (u, v) of a sparse shortcutting into two edges (u, w) and (w, v) in order to reduce the in-degrees of u and v. By carefully choosing vertex w for every edge (u, v), we show that the arboricity can be bounded by $\alpha_{k/2+1}(n)$ when the hop-diameter is k. Roughly speaking, we achieve the same sparsity bound with a twice as large hop-diameter, which is the consequence of splitting each edge into two. Our construction for paths is then readily applicable to HSTs. Using the known HST cover [15], we obtain the following theorem. See Section 4 for details.

▶ **Theorem 5.** *Let k be an even integer and let $\epsilon \in (0, 1/6)$ be an arbitrary parameter. Then, for every positive integer n, every n-point metric with doubling dimension d admits a $(1+\epsilon)$-spanner with hop-diameter k and arboricity $\epsilon^{-O(d)}\alpha_{k/2+1}(n)$.*

This significantly strengthens the construction of Arya et al. [4], by providing a uniformly sparse (rather than just sparse) construction with constant hop-diameter: For hop-diameter k, we transition from sparsity $\epsilon^{-O(d)}\alpha_k(n)$ to arboricity $\epsilon^{-O(d)}\alpha_{k/2+1}(n)$. In particular, we get arboricity $O(\log^* n)$ with a hop-diameter of 6. Recall that, in contrast to arboricity, one cannot achieve similar dependencies on the treewidth.

For general trees, we obtain a k-hop shortcutting with arboricity $O(\log^{12/(k+4)} n)$. While the arboricity is larger than that of ultrametrics (cf. Lemma 16), it is smaller than the treewidth in Theorem 3 by about a factor of k (at a cost of some slack to the exponent of $\log n$). Specifically, when $k = \Theta(\log\log n)$, the arboricity is $O(1)$, while the treewidth is $\Omega(\log\log n)$. Using this result and the known tree cover constructions [10, 11, 5, 20], we obtain low-hop spanners for planar, minor-free, and general metrics with small arboricity. See the full version for details.

2 Preliminaries

Treewidth. A tree decomposition of $G = (V, E)$ is a tree T where each node is associated with a subset of V called a bag. The bags $X_1, \ldots, X_\ell$ satisfy: (i) $\cup_{i=1}^{\ell} X_i = V$, (ii) for every vertex $v \in V$, the collection of bags containing it forms a connected subtree of T, and (iii) for every edge $(u, v) \in E$, there is a bag containing u and v. The width of T is $\max_{i=1}^{\ell} |X_i| - 1$; the treewidth of G is the minimum width among all possible tree decompositions of G. If a graph G contains K_h as a minor, then the treewidth of G is at least $h - 1$.

Arboricity. The arboricity of $G = (V, E)$ is defined as $\max\lceil \frac{|E(H)|}{|V(H)|-1} \rceil$, where the maximum is taken over all subgraphs H of G with at least two vertices.

Spanners. Given a metric space $M_X = (X, \delta)$, which we view as a complete graph $(X, \binom{X}{2})$, a *t-spanner* of M_X is a subgraph $H = (X, E)$ with $E \subseteq \binom{X}{2}$, such that for every two vertices $x, y \in X$, their distance in H is at most $t \cdot \delta(x, y)$. The path $P_{x,y}$ realizing this distance is called a t-spanner path. The *hop-diameter* of a spanner is the minimum k such that between every two vertices $x, y \in X$ there is a t-spanner path with at most k edges.

Tree covers. Given a metric space $M_X = (X, \delta)$, a tree cover of M_X with stretch t is a collection of trees such that for every two vertices $x, y \in X$ we have $\delta(x, y) \le \min_{T \in \mathcal{T}} \delta_T(x, y) \le t \cdot \delta(x, y)$, where $\delta_T(x, y)$ is the distance between x and y in T.

3 Low-treewidth shortcuttings of trees: lower bound

In this section, we prove Theorem 2, restated below for the reader's convenience.

▶ **Theorem 2.** *For every $n \ge 1$, every shortcutting with hop-diameter k for an n-point path must have treewidth:*

- *$t = \Omega(k \log^{2/k} n)$ for even k and $t = \Omega(k(\frac{\log n}{\log\log n})^{2/(k-1)})$ for odd $k \ge 3$, whenever $k \le \frac{2}{\ln(2e)} \ln\log n$;*
- *$t = \Omega((\log\log n)^2/k)$ whenever $k > \frac{2}{\ln(2e)} \ln\log n$.*

Given an arbitrary integer n, our instance is the n-point path. We argue that any shortcutting of the path has a large minor. Due to the inductive nature of our argument, we prove a stronger version of the statement, which considers shortcuttings that could potentially use points outside of the given path. We start with the proof for hop-diameter $k = 2$.

▶ **Lemma 6.** *For every $n \geq 1$ and every n-vertex path L, every shortcutting with hop-diameter 2 has $K_{\lfloor \log n \rfloor + 1}$ as a minor.*

Proof. We prove the claim by complete induction over n. For the base case, we use $n = 1$, where the claim holds vacuously. Let H be a shortcutting for L with hop-diameter 2. Split L into two consecutive parts, L_1 and L_2, of sizes $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$, respectively. Let H_1 and H_2 be subgraphs of H induced on L_1 and L_2, respectively. From the induction hypothesis, H_1 and H_2 have $K_{\log \lfloor n/2 \rfloor + 1}$ and $K_{\log \lceil n/2 \rceil + 1}$ as minors, respectively. Consider the case where every point of L_1 has an edge in H to some point in L_2. Then $K_{\log \lfloor n/2 \rfloor} \cup \{H_2\}$ induces a clique minor of size $\log \lfloor n/2 \rfloor + 1$. Consider the complementary case where L_1 has a point p that does not have an edge in H to any point in L_2. Then, every point in L_2 has a neighbor in L_1 because H has hop-diameter 2. Thus, $K_{\log \lceil n/2 \rceil} \cup \{H_1\}$ induces a clique minor of size $\log \lceil n/2 \rceil + 1$. Hence, the minor size satisfies recurrence $W_2(n) \geq W_2(\lfloor n/2 \rfloor) + 1$, with a base case $W_2(1) = 1$. The solution is given by $W_2(n) = \lfloor \log n \rfloor + 1$. ◀

We next give a proof for even values of k such that $4 \leq k \leq \frac{2}{\ln(2e)} \ln \log n$ in Lemma 7. The proof for odd values is analogous. The proof for $k > \frac{2}{\ln(2e)} \ln \log n$ is given in Lemma 8. We will need some auxiliary claims (Claims 9–11) in the proofs of Lemmas 7 and 8, as well as in the upper bound proof of Theorem 3. These claims are deferred to the end of this section.

▶ **Lemma 7.** *For every $n \geq 1$, every even k such that $4 \leq k \leq \frac{2}{\ln(2e)} \ln \log n$, and every n-vertex path L, every shortcutting with hop-diameter k has treewidth at least $c_1 k \log^{2/k} n - 1$, where c_1 is an absolute constant.*

Proof. We prove the statement by complete induction over n and k. For the base case, we take $n = 1$, and $W_k(1) = 1 > c_1 k \log^{2/k} n - 1$. Otherwise, let ℓ_k be such that $\log \ell_k = (\frac{k}{k-2})^{(k-2)/2} (\log n)^{(k-2)/k}$ for even values of k. (The proof for odd values is similar. There, we choose ℓ_k so that $\log \ell_k = (\frac{k}{k-2})^{(k-2)/2} (\log n / \log \log n)^{(k-2)/k}$.) By Claim 10, we have that $\ell_k \leq \sqrt{n}$, whenever $k \leq \frac{2}{\ln(2e)} \ln \log n$.

Split L into consecutive sets of points $L_1, L_2, \ldots, L_{\ell_k}$ of size $\lfloor n/\ell_k \rfloor$ each and ignore the remaining points. Let H be a shortcutting with hop-diameter k for L. Our goal is to show that the size of a clique minor of H can be lower bounded by the following recurrence.

$$W_k(n) \geq \min(W_{k-2}(\ell_k), W_k(n/(2\ell_k)) + 1) \text{ and } W_k(1) = 1 \tag{1}$$

We say that a point in L_i is *global* if it has an edge to a point outside L_i and *non-global* otherwise. We say that L_i is global if all of its points are global and *non-global* otherwise.

Case 1: Every L_i is non-global. See Figure 1 for an illustration. For every L_i and L_j the path between a non-global point in L_i and a non-global point in L_j must have the first (resp., last) edge inside L_i (resp., L_j). Let H' be obtained from H by contracting each L_i into a single vertex. (Clearly, $H[L_i]$ is connected.) Let L' be the path obtained from L by contracting every L_i into a single point. Then H' is a $(k-2)$-hop spanner of L' with stretch 1. Thus, H' has a minor of size $W_{k-2}(\ell_k) \geq c_1(k-2) \log^{2/(k-2)} \ell_k - 1 = c_1 k \log^{2/k} n - 1$.

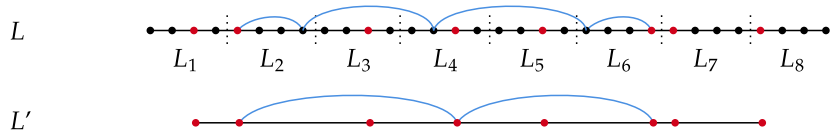


Figure 1 An illustration of Case 1. Every set L_i contains a non-global point. Any k-hop path between two non-global points must have the first and the last edge inside the corresponding sets. By contracting every L_i into a single point, we obtain a new path L'. Any shortcutting with hop-diameter k for L must have hop-diameter $k-2$ for L'. This allows us to use the induction hypothesis for $k-2$. In the illustration above the blue path consisting of 4 hops in L becomes a 2-hop path in L'.

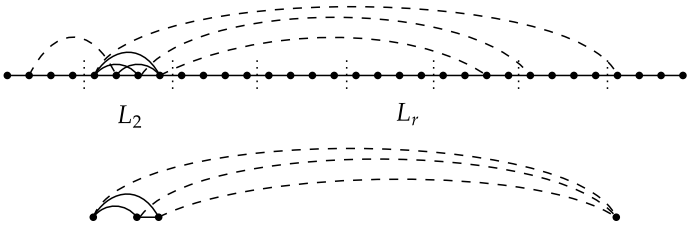


Figure 2 An illustration of Case 2. L_2 is global, meaning that every vertex is adjacent to an edge going outside of L_2. At least half of these edges are adjacent on the vertices in L_r. By contracting L_r and using the clique minor of L_2, we obtain the desired clique minor.

Case 2: Some L_i is global. See Figure 2 for an illustration. Let $\{L_l, L_r\} = L \setminus L_i$, so that L_l (resp., L_r) is on the left (resp., right) of L_i. (Possibly $L_l = \emptyset$ or $L_r = \emptyset$.) At least $|L_i|/2$ points in L_i have edges to either L_l or L_r. Without loss of generality, we assume the latter. Let $L_i' \subseteq L_i$ be the subset of points that have edges to L_r and let H_i be a subgraph of H which is a shortcutting of L_i with hop-diameter k. Inductively, H_i has a clique minor of size at least $W_k(n/(2\ell_k))$. (Since H_i is a 1-spanner, it does not include any point outside of L_i.) Then H_i and L_r are vertex-disjoint (because we are considering 1-spanners) and hence their union has a clique minor of size $W_k(n/(2\ell_k)) + 1$. Thus, $W_k(n)$ satisfies Equation (1), which we lower bound next.

$$
\begin{aligned}
W_k(n) &\geq \min(c_1 k(\log n)^{2/k} - 1, W_k(n/(2\ell_k)) + 1) \\
&\geq \min(c_1 k(\log n)^{2/k} - 1, c_1 k(\log n - \log \ell_k - 1)^{2/k}) \\
&\geq \min\left(c_1 k(\log n)^{2/k} - 1, c_1 k\left(\log n - \left(\frac{k}{k-2}\right)^{\frac{k-2}{2}} \cdot (\log n)^{(k-2)/k} - 1\right)^{2/k}\right) \\
&\geq \min\left(c_1 k(\log n)^{2/k} - 1, c_1 k(\log n)^{2/k} - 1\right)
\end{aligned}
$$

The last inequality follows from Claim 9 by replacing $c_1 = 1/\gamma$, $x = \log n$ and $\alpha = 1$. ◀

▶ **Lemma 8.** *For every $n \geq 1$, every $k > \frac{2}{\ln(2e)} \ln\log n$, every shortcutting with hop-diameter k for an n-vertex path has treewidth at least $c_2(\log\log n)^2/k$, for an absolute constant c_2.*

Proof. We set ℓ_k so that $\log\log \ell_k = \sqrt{\frac{k-2}{k}} \log\log n$. (We use $\log(\cdot) \coloneqq \log_2(\cdot)$.) For the clarity of exposition, we ignore the rounding issues. We note that $1 \leq \ell_k \leq n$. Using the same argument as in Lemma 7, we have $W_k(n) \geq \min(W_{k-2}(\ell_k), W_k(n/(2\ell_k)) + 1)$. We prove the lemma by induction, where the base case is Lemma 7 whenever $k < \frac{2}{\ln(2e)} \ln\log n$. Our goal is to prove that $W_{k-2}(\ell_k) \geq c_2(\log\log n)^2/k$ and $W_k(n/(2\ell_k)) + 1 \geq c_2(\log\log n)^2/k$. For the first inequality we distinguish two cases. If $k - 2 \leq \frac{2}{\ln(2e)} \ln\log(\ell_k)$, then by Lemma 7 we have

$$\begin{aligned} W_{k-2}(\ell_k) &\geq c_1(k-2)\log^{\frac{2}{k-2}} \ell_k - 1 \geq c_1(k-2)\log^{\frac{\ln(2e)}{\ln\log \ell_k}} \ell_k - 1 = 2ec_1(k-2) - 1 \\ &\geq ec_1 k - 1 \geq ec_1 \cdot \left(\frac{2}{\ln(2e)}\right)^2 \frac{(\ln\log \ell_k)^2}{k} - 1 \geq c_2 \frac{(\log\log \ell_k)^2}{k-2} \end{aligned}$$

The penultimate inequality holds because $k \geq \frac{2}{\ln(2e)} \ln\log(n) \geq \frac{2}{\ln(2e)} \ln\log(\ell_k)$. The last inequality holds for a proper choice of constant c_2. If $k - 2 > \frac{2}{\ln(2e)} \ln\log(\ell_k)$, we have $W_{k-2}(\ell_k) \geq c_2 \frac{(\log\log \ell_k)^2}{k-2}$ by the induction hypothesis. Hence, in both cases, we have:

$$W_{k-2}(\ell_k) \geq c_2 \cdot \frac{(\log\log \ell_k)^2}{k-2} = c_2 \cdot \frac{\left(\sqrt{\frac{k-2}{k}} \log\log n\right)^2}{k-2} = c_2 \cdot \frac{(\log\log n)^2}{k}$$

For the second inequality, let $x = \log n$. We have $\log \ell_k = x^{\sqrt{(k-2)/k}}$. Since $\frac{n}{2\ell_k} < n$, we have that $k > \frac{2}{\ln(2e)} \ln\log \frac{n}{2\ell_k}$ and the induction hypothesis gives the following.

$$W_k\left(\frac{n}{2\ell_k}\right) + 1 \geq \frac{c_2 \left(\log\log\left(\frac{n}{2\ell_k}\right)\right)^2}{k} + 1 = \frac{c_2 \log^2\left(x - x^{\sqrt{(k-2)/k}} - 1\right)}{k} + 1$$

To show that the right-hand side is at least $c_2 \log^2(x)/k$, it suffices to show the following:

$$\log^2\left(x - x^{\sqrt{(k-2)/k}} - 1\right) + \frac{k}{c_2} \geq \log^2 x \tag{2}$$

From Claim 11 we have $x^{\sqrt{(k-2)/k}} \leq x - \frac{x \ln x}{k} + \frac{x \ln^2 x}{k^2}$.

$$x - x^{\sqrt{(k-2)/k}} - 1 \geq x - \left(x - \frac{x \ln x}{k} + \frac{x \ln^2 x}{k^2}\right) - 1 = \frac{x \ln x}{k}\left(1 - \frac{\ln x}{k}\right) - 1 > \frac{x \ln x}{10k} - 1$$

The last inequality holds because $k > \frac{2}{\ln(2e)} \ln(x)$. We next consider two cases. If $\frac{x \ln x}{10k} < 2$, then $\frac{k}{c_2} > \frac{x \ln x}{20c_2} \geq \log^2 x$ and Equation (2) is proved. Otherwise, we proceed as follows.

$$\begin{aligned} \log^2\left(\frac{x \ln x}{10k} - 1\right) + \frac{k}{c_2} &\geq \log^2\left(\frac{x \ln x}{20k}\right) + \frac{k}{c_2} = \left(\log x + \log\frac{\ln x}{20k}\right)^2 + \frac{k}{c_2} \\ &= \log^2 x + 2(\log x)\log\frac{\ln x}{20k} + \log^2\frac{\ln x}{20k} + \frac{k}{c_2} \geq \log^2 x \end{aligned}$$

The last inequality holds for any $c_2 \leq 1/10$. ◀

We conclude the section by proving the auxiliary claims used in the lemmas above.

▷ Claim 9. There is an absolute constant γ such that for $\alpha \in \{0,1\}$, every integer $k \geq 4$ and every $x > 1$ where the expression is defined, it holds

$$\frac{2}{k} < x^{2/k} - \left(x - \left(\frac{k}{k-2}\right)^{(k-2)/2} \cdot x^{(k-2)/k} - \alpha\right)^{2/k} < \frac{\gamma}{k}$$

Proof. We rewrite the expression as follows.

$$x^{2/k}\left(1 - \left(1 - \left(\frac{k}{k-2}\right)^{(k-2)/2} \cdot x^{-2/k} - \alpha x^{-1}\right)^{2/k}\right) \tag{3}$$

Using Maclaurin expansion, we have that $(1+y)^{2/k} = 1 + \frac{2}{k}y + \frac{2-k}{k^2} \cdot (1+\zeta)^{\frac{2}{k}-2} \cdot y^2$, where ζ is a number between 0 and y. We set $y = -\left(\frac{k}{k-2}\right)^{(k-2)/2} \cdot x^{-2/k} - \alpha x^{-1}$.

$$\left(1 - \left(\frac{k}{k-2}\right)^{(k-2)/2} \cdot x^{-2/k} - \alpha x^{-1}\right)^{2/k} = \tag{4}$$

$$1 - \frac{2}{k}\left(\left(\frac{k}{k-2}\right)^{\frac{k-2}{2}} x^{-\frac{2}{k}} + \alpha x^{-1}\right) - \frac{k-2}{k^2}(1+\zeta)^{\frac{2}{k}-2}\left(\left(\frac{k}{k-2}\right)^{\frac{k-2}{2}} \cdot x^{-\frac{2}{k}} + \alpha x^{-1}\right)^2 \tag{5}$$

Plugging Equation (4) into Equation (3), we obtain the following.

$$x^{2/k}\left(1 - \left(1 - \left(\frac{k}{k-2}\right)^{\frac{k-2}{2}} \cdot x^{-2/k} - \alpha x^{-1}\right)^{2/k}\right) =$$

$$\frac{2}{k}\left(\left(\frac{k}{k-2}\right)^{\frac{k-2}{2}} + \alpha x^{-\frac{k-2}{k}}\right) + \frac{k-2}{k^2}(1+\zeta)^{\frac{2}{k}-2}\left(\left(\frac{k}{k-2}\right)^{\frac{k-2}{2}} \cdot x^{-1/k} + \alpha x^{-\frac{k-1}{k}}\right)^2$$

The lower bound in Claim 9 holds because $-1 < y < \zeta < 0$ and $x > 1$. Next we prove the upper bound.

$$\frac{2}{k}\left(\left(\frac{k}{k-2}\right)^{\frac{k-2}{2}} + \alpha x^{-\frac{k-2}{k}}\right) + \frac{k-2}{k^2}(1+\zeta)^{\frac{2}{k}-2}\left(\left(\frac{k}{k-2}\right)^{\frac{k-2}{2}} \cdot x^{-1/k} + \alpha x^{-\frac{k-1}{k}}\right)^2 <$$

$$\frac{2(e + x^{-(k-2)/k}) + (ex^{-1/k} + x^{-(k-1)/k})^2}{k}$$

The right-hand side is decreasing with x in the whole domain and we can upper bound it by taking $x = 1$.

$$\frac{2(e + x^{-(k-2)/k}) + (ex^{-1/k} + x^{-(k-1)/k})^2}{k} < \frac{(e+1)(e+3)}{k}$$

Letting $\gamma = (e+1)(e+3)$, the upper bound from Claim 9 follows. ◁

▷ Claim 10. For every $3 \leq k \leq \frac{2}{\ln(2e)} \ln\log n$, it holds that $(\frac{k}{k-2})^{(k-2)/2}(\log n)^{(k-2)/k} \leq (\log n)/2$.

Proof. We have that $k \leq \frac{2}{\ln(2e)} \ln\log n$. Rearranging the last inequality, we have that $e(\log n)^{(k-2)/k} \leq (\log n)/2$. The proof is completed by observing that $(\frac{k}{k-2})^{(k-2)/2}$ is monotonically increasing for $k \geq 3$ and $\lim_{k\to\infty}(\frac{k}{k-2})^{(k-2)/2} = e$. ◁

▷ Claim 11. For every $x \geq 1$ and $k \geq 4$, $x^{\sqrt{(k-2)/k}} \leq x - \frac{x \ln x}{k} + \frac{x \ln^2 x}{k^2}$

Proof. Let $p := 1/k$. Then, $x^{\sqrt{(k-2)/k}} = x^{\sqrt{1-2p}}$. Using Taylor series around $p = 0$, we have that $x^{\sqrt{1-2p}} = x - px \ln x + R(p)$. The term $R(p)$ has the following form for $0 < \xi < p$.

$$R(p) = \frac{p^2}{2} \cdot x^{\sqrt{1-2\xi}} \ln x \left(\frac{\ln x}{1-2\xi} - (1-2\xi)^{-3/2} \right) \leq \frac{p^2}{2} \cdot x^{\sqrt{1-2\xi}} \ln x \cdot \frac{\ln x}{1-2\xi} \leq p^2 x \ln^2 x$$

To complete the proof, we replace p by $1/k$. ◁

4 Low-arboricity shortcuttings

In this section, we prove Theorem 5.

▶ **Theorem 5.** *Let k be an even integer and let $\epsilon \in (0, 1/6)$ be an arbitrary parameter. Then, for every positive integer n, every n-point metric with doubling dimension d admits a $(1+\epsilon)$-spanner with hop-diameter k and arboricity $\epsilon^{-O(d)} \alpha_{k/2+1}(n)$.*

We first describe the shortcutting on paths (Section 4.1) and then adapt it to HSTs via an analogous argument (Section 4.2). Instead of directly arguing about the arboricity, we show that the shortcutting can be oriented so that it has a bounded in-degree. The connection is provided by the following fact.

▶ **Fact 12.** *If the edges of a graph $G = (V, E)$ can be oriented such that the maximum in-degree of every vertex is at most d, then the arboricity of G is at most $d + 1$.*

4.1 Paths

The main intuition of this part of the proof is given in Figure 3. We shall use a modification of the following well-known result.

▶ **Theorem 13** (Cf. [3, 7, 22])**.** *For every $n \geq 2$ and $k \geq 2$, every n-point tree metric admits a shortcutting with hop-diameter k and $O(n\alpha_k(n))$ edges.*

We next prove a slightly modified version of this theorem.

▶ **Lemma 14.** *Let $n \geq 1$ and $k \geq 1$ be two arbitrary integers. Let L be a line metric induced by a set of n points on a line so that between every two points there are $\alpha_k(n)$ Steiner points for $k \geq 2$ and n^2 points for $k = 1$. Then, L admits a Steiner 1-spanner with hop-diameter $2k$ such that the Steiner points belong to S and every vertex in $L \cup S$ has a constant in-degree.*

Proof. We prove the lemma by induction over k. For $k = 1$, we interconnect the vertices in L by a clique. Consider an arbitrary clique edge (u, v) and split it into two using a Steiner point w. Orient the edges (u, w) and (w, v) into w. By using a fresh Steiner point for every clique edge, we obtain the guarantees from the statement. For $k = 2$, we take a central vertex c in L and connect it to every other point in L; orient the edges outwards from c. Proceed recursively with the two halves. This way we obtain a 1-spanner for L with hop-diameter 2. Denote by E the edge set of this spanner. The depth of the recursion in the construction is $O(\log n)$, and the size of S is $n\alpha_2(n) = n \log n$. Every vertex in L has in-degree 1 per recursion level. We can split every such edge into two using a fresh Steiner point for each recursion level. This concludes the proof for $k = 2$.

Consider now an arbitrary k. Divide L into intervals of size $\alpha_{k-2}(n)$ using $n/\alpha_{k-2}(n)$ cut vertices. Denote by C the set of cut vertices and invoke the induction hypothesis on C with parameter $k - 2$. Let E' be the obtained set of edges. Let E_C be obtained by connecting

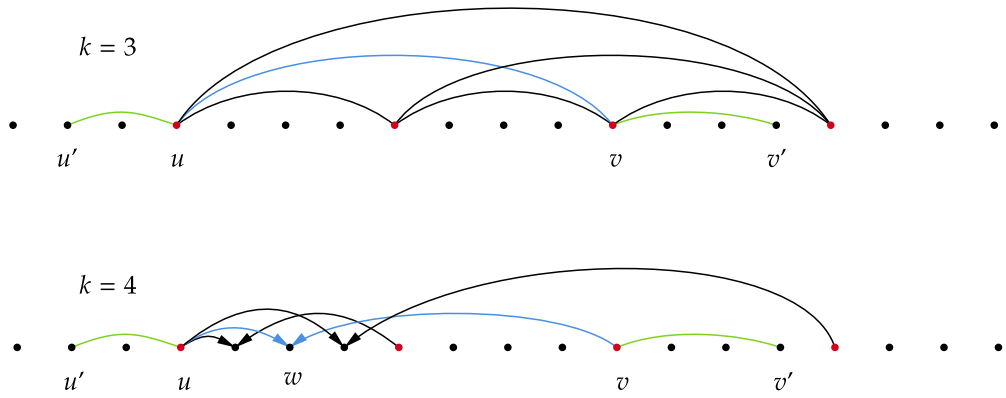


Figure 3 An illustration of the shortcutting with hop-diameter 4 and arboricity $O(\log\log n)$. We start by a well-known 3-hop construction [2, 7]. Split the path into subpaths using $\sqrt{n}$ cut vertices (red) so that every subpath contains $\sqrt{n}$ vertices. The shortcutting is obtained by: *(i)* interconnecting the cut vertices by a clique, *(ii)* connecting every cut vertex to the points in its two adjacent subpaths (in the illustration, only two such edges, (u', u) and (v, v'), are shown in green), and *(iii)* applying the recursive construction to each of the subpaths. This construction readily gives hop-diameter 3 (see, e.g., the path between u' and v'), but it suffers from having arboricity $\Omega(\sqrt{n})$ due to the clique on the cut vertices. To remedy this, we split every clique edge (u, v) into two edges, (u, w) and (w, v). This allows us to reduce the in-degree of every point to $O(1)$ per recursion level, at a cost of increasing the hop-diameter to 4. Overall, the number of recursion levels is $O(\log\log n)$, yielding the desired bound on the arboricity.

every cut vertex to every point in the two neighboring intervals; the edges are oriented out of E_C. Proceed recursively with parameter k on each of the intervals.

To analyze the in-degree, we observe that the depth of the recursion with parameter k is $O(\alpha_k(n))$, which coincides with the number of Steiner vertices between every two consecutive points in L. One level of recursion contributes a constant in-degree to each vertex in the construction. This means that we can split all such vertices into two and use a fresh Steiner point at each recursion level. This concludes the proof. ◀

▶ **Proposition 15.** *For every $n \geq 1$ and every even $k \geq 2$, every n-point path admits a shortcutting with hop-diameter k and arboricity $O(\alpha_{k/2+1}(n))$.*

Proof. Let L_n be an arbitrary line metric. For an integer $k' \geq 2$, we describe a construction of a shortcutting H for L_n with hop-diameter $4k' - 2$ and arboricity $O(\alpha_{2k'}(n))$.

Consider a set of $n' = n/\alpha_{2k'-2}(n)$ equally-spaced *cut vertices* dividing L_n into subpaths consisting of $\alpha_{2k'-2}(n)$ points. To construct the shortcutting H, we connect every cut vertex to all the vertices in its two neighboring intervals. Denote the corresponding edge by E_C. Let C be the set of cut vertices. Let E' be obtained by invoking Lemma 14 with parameter $2k' - 2$ on C using $L_n \setminus C$ as Steiner points. Proceed recursively with each of the intervals.

To analyze the arboricity, we will show that the edges in E_C and E' can be oriented so that the in-degree of every vertex is constant. Orient every edge in E_C so that it goes out of the corresponding cut vertex. Since every interval is adjacent to at most two cut vertices, the in-degree of every point with respect to E_C is at most 2. By Lemma 14, the edges in E' have a constant in-degree on L_n. In conclusion, E_C and E' contribute $O(1)$ to the in-degree of each vertex in L_n. The number of recursion levels $\ell(n)$ satisfies the recurrence $\ell(n) = \ell(\alpha_{2k'-2}(n)) + O(1)$, which has solution $\ell(n) = \alpha_{2k'}(n)$. Every recursion

level contributes $O(1)$ to the in-degree of vertices, meaning that the overall in-degree in H is $O(\alpha_{2k'}(n))$. The hop-diameter of H is $2 + 2 \cdot (2k' - 2) = 4k' - 2$. This concludes the description of a shortcutting with hop-diameter $4k' - 2$ and arboricity $O(\alpha_{2k'}(n)$.

Note that we have only shown how to get a tradeoff of hop-diameter $4k' - 2$ and arboricity $O(\alpha_{2k'}(n))$. We could similarly get a construction with hop-diameter $4k'$ and arboricity $O(\alpha_{2k'+1}(n))$ for a parameter $k' \geq 1$. Specifically, we divide L_n into intervals of size $\alpha_{2k'-1}(n)$. The cut vertices are interconnected using Lemma 14 with hop-diameter $2k' - 1$. The hop-diameter is $2 + 2(2k' - 1) = 4k'$ and the arboricity is $O(\alpha_{2k'+1}(n))$ using a similar argument. ◀

4.2 Ultrametrics and doubling metrics

A metric (X, d_X) is an *ultrametric* if it satisfies a strong form of the triangle inequality: for every x, y, z, $d_X(x, z) \leq \max\{d_X(x, y), d_X(y, z)\}$. It is well known that an ultrametric can be represented as a *hierarchical well-separated tree* (HST). More precisely, an (s, Δ)-*HST* is a tree T where *(i)* each node v is associated with a label Γ_v such that $\Gamma_v \geq s \cdot \Gamma_u$ whenever u is a child of v and *(ii)* each internal node v has at most Δ children. Parameter s is called the *separation* while Δ is called the *degree* of the HST. Let L be the set of leaves of T. The labels of internal nodes in T induce a metric (L, d_L) on the leaves, called leaf-metric, where for every two leaves $u, v \in L$, $d_L(u, v) = \Gamma_{\text{lca}(u,v)}$ where $\text{lca}(u, v)$ is the lowest common ancestor of u and v. It is well-known, e.g., [6], that (L, d_L) is an ultrametric, and that any ultrametric is isomorphic to the leaf-metric of an HST.

Chan and Gupta [9] showed that any $(1/\epsilon, 2)$-HST can be embedded into the line metric with (worst-case) distortion $1 + O(\epsilon)$. Therefore, by applying Proposition 15, we obtain a $(1 + O(\epsilon))$-spanner with hop diameter k and arboricity $O(\alpha_{k/2+1}(n))$ for $(1/\epsilon, 2)$-HSTs. In our setting, we are interested in large-degree (s, Δ)-HSTs where $s = 1/\epsilon$ and $\Delta = \text{poly}(1/\epsilon)$; the embedding result by Chan and Gupta [9] no longer holds for these HSTs. Instead, we directly apply our technique for the line metric to get a $(1 + O(\epsilon))$-spanner with low-hop diameter. We describe the modifications of Proposition 15 required to prove Lemma 16.

▶ **Lemma 16.** *Let $\epsilon \in (0, 1)$, $\Delta > 0$ be parameters, and k be an even positive integer. Then, any $(\Theta(1/\epsilon), \Delta)$-HST with n leaves admits a $(1 + \epsilon)$-spanner with hop-diameter k and arboricity $O(\alpha_{k/2+1}(n))$.*

Proof. Let T be the $(1/\epsilon, \Delta)$-HST and let M_T be the metric induced by T. For an integer $k' \geq 2$, we describe a construction of a $(1 + O(\epsilon))$-spanner for M_T with hop-diameter $4k' - 2$ and arboricity $O(\alpha_{2k'}(n))$. The construction is similar to that in Proposition 15.

Let C be the set of internal nodes of T, called *cut vertices*, such that the subtrees rooted at these nodes has size $\alpha_{2k'-2}(n)$. The number of cut vertices is $|C| \leq n/\alpha_{2k'-2}(n)$. First, connect every cut vertex to all of its descendants in T and let the corresponding set of edges be E_C. Next, let E be the set of edges interconnecting the cut vertices using Theorem 13 with hop-diameter $2k' - 2$ and $O(n'\alpha_{2k'-2}(n')) = O(n)$ edges. We construct a set E' by subdividing every edge $(u, v) \in E$ into two edges using a vertex, say x, in the subtree rooted at u. The spanner H is obtained using the edges in E_C and those in E'. Finally, the recursive construction is applied to each subtree rooted at a vertex in C. This concludes the description of the recursive construction of H. The same argument in Proposition 15 implies that the arboricity is $O(\alpha_{2k'}(n))$. The stretch is $(1 + O(\epsilon))$ since the path $u \to x \to v$ between two cut vertices u and v has stretch $(1 + O(\epsilon))$. ◀

To construct a low-hop spanner with small arboricity for doubling metrics (Theorem 5), we will rely on the ultrametric cover by Filtser and Le [15]. Following their notation, for a given metric space (X, d_X), we say that a collection $\mathcal{T}$ of at most τ different (s, Δ)-HSTs

such that (i) for every HST $T \in \mathcal{T}$, points in X are leaves in T, and (ii) for every two points $x, y \in X$, $d_X(x,y) \leq d_T(x,y)$ for every $T \in \mathcal{T}$, and there exists a tree $T_{xy} \in \mathcal{T}$ such that $d_X(x,y) \leq \rho \cdot d_{T_{xy}}(x,y)$.

▶ **Theorem 17** (Cf. Theorem 3.4 in [15]). *For every $\epsilon \in (0, 1/6)$, every metric with doubling dimension d admits an $(\epsilon^{-O(d)}, 1+O(\epsilon), 1/\epsilon, \epsilon^{-O(d)})$-ultrametric cover.*

Let $\mathcal{T}$ be the $(\epsilon^{-O(d)}, 1+O(\epsilon), 1/\epsilon, \epsilon^{-O(d)})$-ultrametric cover in Theorem 17 for the input doubling metric. We construct the spanner from Lemma 16 on each of $(1/\epsilon, \epsilon^{-O(d)})$-HST in $\mathcal{T}$ and take the union of them. This concludes the proof of Theorem 5.

5 Routing schemes

A compact routing scheme is a distributed algorithm that uses local information (routing tables and unique labels) to deliver packets between any source and destination node in a network. In this section, we prove Theorem 4.

▶ **Theorem 4.** *For every n and every n-point metric with doubling dimension d, there is a 3-hop routing scheme with stretch $(1+\epsilon)$ that uses $\epsilon^{-\tilde{O}(d)} \cdot \log^2 n / \log\log n$ bits per vertex.*

Our starting point is the routing scheme for the trees.

▶ **Theorem 18.** *For every n and every n-vertex tree T, there is a 3-hop routing scheme in the fixed-port model for the metric M_T induced by T with stretch 1 that uses $O(\log^2 n / \log\log n)$ bits per vertex.*

Proof. Given an n-vertex tree T, our routing scheme operates on top of a shortcutting with hop-diameter 3 and treewidth $O(\log n / \log\log n)$. We start by sketching this construction. Note that the routing scheme does not rely on the treewidth bound itself, but on the fact that the edges of the shortcutting can be oriented so that every vertex has an in-degree bounded by $O(\log n / \log\log n)$.

Let X be a set of vertices of T such that: *(i)* every component of $T \setminus X$ is of size $O(n \log\log n / \log n)$, *(ii)* the size of X is $O(\log n / \log\log n)$, *(iii)* every component of $T \setminus X$ has at most 2 edges adjacent to X, and *(iv)* if a component C has 2 edges adjacent to some $x, y \in X$, then x and y are in ancestor/descendant relation. Connect the vertices of X by a clique and do the following for every subtree T' in $T \setminus X$. Let u and v be two vertices from X adjacent to T'. Connect u and v to every vertex in T' and proceed recursively with T'. When $n \leq 3$, connect the vertices by a clique. This concludes the description of the shortcutting. The constructions of low-treewidth shortcuttings for general values of hop-diameter are more technical (and not needed for this section). See the full version for the details.

Our routing scheme is constructed on top of a 3-hop shortcutting H_3 described above. For a vertex $u \in T$, let table(u) denote its routing table and label(u) its label. First, assign a unique identifier $ID(u) \in \{1, \ldots, n\}$ to every vertex u in T and add it to table(u) and label(u). Equip the routing table and a label of every vertex $u \in T$ with an ancestor label anc(u) as in [1]. Using this ancestor labeling scheme, we can determine, given two vertices u and v, whether they are in ancestor-descendant relation, and if so, whether u is the ancestor of v or not. This adds $O(\log n)$ bits of memory per vertex.

Recall the recursive construction of the shortcutting. Assign to each recursive call a unique integer r_T. For every vertex in $u \in X$, add to table(u) the information consisting of $C(u) = \langle r_T, \{\langle ID(v), \text{port}(u,v), \text{anc}(v)\rangle \mid v \in X \setminus \{x\}\}\rangle$. The memory occupied per every vertex in X is $O(\log^2 n / \log\log n)$. (Note that the construction of H_3 guarantees that

every vertex belongs to such a clique exactly once across all the recursive calls, meaning that table(u) contains only one such $C(u)$.) Let T' be a subtree in $T \setminus X$. Let u and v be two vertices from X to which T' has outgoing edges. For every vertex $x \in T'$, add to table(x) the following: $\langle r_T, ID(u), \text{port}(x,u)\rangle$ and $\langle r_T, ID(v), \text{port}(x,v)\rangle$. Similarly, add to label(x) the following: $\langle r_T, ID(u), \text{port}(u,x)\rangle$ and $\langle r_T, ID(v), \text{port}(v,x)\rangle$. This information takes $O(\log n)$ bits per recursive call r_T. The construction proceeds recursively with T'; the number of recursive calls every vertex participates in is at most $O(\log n/\log\log n)$.

Next we describe the routing algorithm. Let u be the source and v be the destination. First, check if $C(u)$ contains routing information leading directly to v. In this case, the algorithm outputs port(u,v) and the routing is complete. (This case happens when u and v are in the same clique during the construction.) Otherwise, go over table(u) and label(v) and find the last recursive call r_T which is common to both u and v. Next, consider label(v) and the two entries consisting $\langle r_T, ID(v_1), \text{port}(v_1,v)\rangle$ and $\langle r_T, ID(v_2), \text{port}(v_2,v)\rangle$, corresponding to r_T. If v_1 and v_2 are in $C(u)$, use anc(u), anc(v_1), anc(v_2), and anc(v) to decide whether to output port(u,v_1) or port(u,v_2). (This case happens when u, v_1, and v_2 are in X in the recursive call r_T and v is not in it.) Finally, let $\langle r_T, ID(u_1), \text{port}(u,u_1)\rangle$ and $\langle r_T, ID(u_2), \text{port}(u,u_2)\rangle$ be the two entries corresponding to r_T in table(u). Use anc(u) and anc(v) to decide whether to output port(u,u_1) or port(u,u_2). (This case happens when u is not in X.) ◀

In what follows, we extend the routing result for tree metrics to the metrics with doubling dimension d. We first construct a tree cover using the tree cover theorem from [13].

▶ **Theorem 19** ([13]). *Given a point set P in a metric of constant doubling dimension d and any parameter $\epsilon \in (0,1)$, there exists a tree cover with stretch $(1+\epsilon)$ and $\epsilon^{-\tilde{O}(d)}$ trees. Furthermore, every tree in the tree cover has maximum degree bounded by $\epsilon^{-O(d)}$.*

We use this specific tree cover theorem, since the authors also provide an algorithm for determining the "distance-preserving tree" given the labels of any two metric points.

▶ **Lemma 20** ([13]). *Let $\epsilon \in (0,1)$. Let $T = \{T_1, \ldots, T_k\}$ be the tree cover for P constructed by Theorem 19, where $k = \epsilon^{-\tilde{O}(d)}$. There is a way to assign $\epsilon^{-\tilde{O}(d)} \log n$-bit labels to each point in P so that, given the labels of two vertices x and y, we can identify an index i such that tree T_i is a "distance-approximating tree" for u and v; that is, $\delta_{Ti}(x,y) \le (1+\epsilon)\delta_P(x,y)$. This decoding can be done in $O(d \cdot \log(1/\epsilon))$ time.*

We equip each tree in the cover with the stretch-1 routing scheme from Theorem 18. This consumes overall $\epsilon^{-\tilde{O}(d)} \log^2 n/\log\log n$ bits per point in P. In addition, we add $\epsilon^{-\tilde{O}(d)} \log n$-bit labels to each point in P as stated in Lemma 20. Given two points $x, y \in P$, we first employ the algorithm from Lemma 20 to find the tree in which the routing should proceed. Then, the routing proceeds on that specific tree using the routing algorithm from Theorem 18. This concludes the proof of Theorem 4.

References

1 Serge Abiteboul, Stephen Alstrup, Haim Kaplan, Tova Milo, and Theis Rauhe. Compact labeling scheme for ancestor queries. *SIAM J. Comput.*, 35(6):1295–1309, 2006. doi:10.1137/S0097539703437211.

2 Noga Alon and Baruch Schieber. Optimal preprocessing for answering on-line product queries. Technical Report TR 71/87, Tel Aviv University, 1987.

3 Noga Alon and Baruch Schieber. Optimal preprocessing for answering on-line product queries. *CoRR*, abs/2406.06321, 2024. doi:10.48550/arXiv.2406.06321.

4 S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: Short, thin, and lanky. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 489–498, 1995.

5 Yair Bartal, Arnold Filtser, and Ofer Neiman. On notions of distortion and an almost minimum spanning tree with constant average distortion. *J. Comput. Syst. Sci.*, 105:116–129, 2019. preliminary version published in SODA 2016. doi:10.1016/j.jcss.2019.04.006.

6 Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. Some low distortion metric ramsey problems. *Discrete & Computational Geometry*, 33(1):27–41, July 2004. doi:10.1007/s00454-004-1100-z.

7 Hans L. Bodlaender, Gerard Tel, and Nicola Santoro. Trade-offs in non-reversing diameter. *Nord. J. Comput.*, 1(1):111–134, 1994.

8 Kevin Buchin, Carolin Rehs, and Torben Scheele. Geometric spanners of bounded tree-width. In *41st International Symposium on Computational Geometry (SoCG 2025)*, pages 26:1–26:15, 2025. doi:10.4230/LIPIcs.SoCG.2025.26.

9 H. T.-H. Chan and A. Gupta. Small hop-diameter sparse spanners for doubling metrics. In *Proc. of 17th SODA*, pages 70–78, 2006.

10 H. Chang, J. Conroy, H. Le, L. Milenković, S. Solomon, and C. Than. Covering planar metrics (and beyond): $O(1)$ trees suffice. In *The 64th Annual Symposium on Foundations of Computer Science*, FOCS '23, pages 2231–2261, 2023. doi:10.1109/FOCS57990.2023.00139.

11 H. Chang, J. Conroy, H. Le, L. Milenković, S. Solomon, and C. Than. Shortcut partitions in minor-free graphs: Steiner point removal, distance oracles, tree covers, and more. In *The 2024 Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '24, pages 5300–5331, 2024. doi:10.1137/1.9781611977912.191.

12 Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenkovic, Shay Solomon, and Cuong Than. Optimal euclidean tree covers. In *SoCG*, volume 293 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SOCG.2024.37.

13 Hsien-Chih Chang, Jonathan Conroy, Hung Le, Shay Solomon, and Cuong Than. Light tree covers, routing, and path-reporting oracles via spanning tree covers in doubling graphs. In *STOC*, pages 2257–2268. ACM, 2025. doi:10.1145/3717823.3718312.

14 Bernard Chazelle. Computing on a free tree via complexity-preserving mappings. *Algorithmica*, 2(1):337–361, 1987. doi:10.1007/BF01840366.

15 Arnold Filtser and Hung Le. Locality-sensitive orderings and applications to reliable spanners. In *STOC*, pages 1066–1079. ACM, 2022. doi:10.1145/3519935.3520042.

16 Arnold Filtser and Hung Le. Low treewidth embeddings of planar and minor-free metrics. In *FOCS*, pages 1081–1092. IEEE, 2022. doi:10.1109/FOCS54457.2022.00105.

17 E. Fox-Epstein, P. N. Klein, and A. Schild. Embedding planar graphs into low-treewidth graphs with applications to efficient approximation schemes for metric problems. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 1069–1088, 2019. doi:10.1137/1.9781611975482.66.

18 Omri Kahalon, Hung Le, Lazar Milenkovic, and Shay Solomon. Can't see the forest for the trees: Navigating metric spaces by bounded hop-diameter spanners. In Alessia Milani and Philipp Woelfel, editors, *Proc. PODC*, pages 151–162. ACM, 2022. doi:10.1145/3519270.3538414.

19 H. Le. Shortcutting trees, 2023. URL: https://minorfree.github.io/tree-shortcutting/.

20 Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society*, 9(2):253–275, 2007.

21 G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

22 Shay Solomon. Sparse euclidean spanners with tiny diameter. *ACM Trans. Algorithms*, 9(3):28:1–28:33, 2013. doi:10.1145/2483699.2483708.

23 Robert Endre Tarjan. Applications of path compression on balanced trees. *Journal of the ACM (JACM)*, 26(4):690–715, 1979. doi:10.1145/322154.322161.

24 Andrew Chi-Chih Yao. Space-time tradeoff for answering range queries (extended abstract). In *STOC*, pages 128–136. ACM, 1982. doi:10.1145/800070.802185.

Approximating Euclidean Shallow-Light Trees

Hung Le
University of Massachusetts Amherst, MA, USA

Shay Solomon
Tel Aviv University, Israel

Cuong Than
University of Massachusetts Amherst, MA, USA

Csaba D. Tóth
California State University Northridge, Los Angeles, CA, USA
Tufts University, Medford, MA, USA

Tianyi Zhang
State Key Laboratory for Novel Software Technology, Nanjing University, China

Abstract

For a weighted graph $G = (V, E, w)$ and a designated source vertex $s \in V$, a spanning tree that simultaneously approximates a *shortest-path tree* w.r.t. source s and a *minimum spanning tree* is called a *shallow-light tree (SLT)*. Specifically, an (α, β)-SLT of G w.r.t. $s \in V$ is a spanning tree of G with *root-stretch* α (preserving all distances between s and all other vertices up to a factor of α) and *lightness* β (its *weight* is at most β times the weight of a minimum spanning tree of G).

It was shown in the early 1990s that (1) for any graph, any source, and any $\epsilon > 0$, there is a $(1+\epsilon, O(1/\epsilon))$-SLT, and (2) there *exist* graphs for which $\beta = \Omega(1/\epsilon)$ for any $(1+\epsilon, \beta)$-SLT.

The focus of this work is on SLTs in *low-dimensional Euclidean spaces*, which are of special interest for some applications of SLTs, in geometric network optimization problems. The aforementioned existential lower bound applies to Euclidean plane, as well. It was shown more than a decade ago that (1) by using *Steiner points*, one can reduce the lightness bound from $O(1/\epsilon)$ to $O(\sqrt{1/\epsilon})$, and (2) there exist point sets in the plane for which $\beta = \Omega(\sqrt{1/\epsilon})$ for any Steiner $(1+\epsilon, \beta)$-SLT.

These *tight existential bounds* for the Euclidean case yield *approximation factors* of $O(1/\epsilon)$ and $O(\sqrt{1/\epsilon})$ on the minimum weight of any non-Steiner and Steiner tree with root-stretch $1+\epsilon$, respectively. Despite the large body of work on SLTs, the basic question of whether a better approximation algorithm exists was left untouched to date, and this holds in any graph family. This paper makes a first nontrivial step towards resolving this question by presenting two *bicriteria approximation* algorithms. For any $\epsilon > 0$, a set P of n points in constant-dimensional Euclidean space and a source $s \in P$, our first (respectively, second) algorithm returns, in $O(n \log n \cdot \mathrm{polylog}(\epsilon^{-1}))$ time, a non-Steiner (resp., Steiner) tree with root-stretch $1 + O(\epsilon \log \epsilon^{-1})$ and weight at most $O(\mathsf{opt}_\epsilon \cdot \log^2 \epsilon^{-1})$ (resp., $O(\mathsf{opt}_\epsilon \cdot \log \epsilon^{-1})$), where opt_ϵ denotes the minimum weight of a non-Steiner (resp., Steiner) tree with root-stretch $1+\epsilon$.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners; Theory of computation → Routing and network design problems; Mathematics of computing → Graph algorithms

Keywords and phrases geometric network design, optimization, shallow-light tree, Steiner point

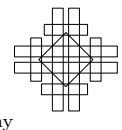
Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.71

Related Version *Full Version*: https://arxiv.org/abs/2512.10797

Funding *Hung Le*: Supported by the NSF CAREER award CCF-2237288, the NSF grants CCF-2517033 and CCF-2121952, a Google Research Scholar Award.
Shay Solomon: Supported by the European Union (ERC, DynOpt, 101043159). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can



© Hung Le, Shay Solomon, Cuong Than, Csaba D. Tóth, and Tianyi Zhang;
licensed under Creative Commons License CC-BY 4.0





be held responsible for them. Shay Solomon is also funded by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, and the United States National Science Foundation (NSF).
Cuong Than: Supported by the NSF CAREER award CCF-2237288, the NSF grants CCF-2517033 and CCF-2121952, a Google Research Scholar Award, and a Google Ph.D. Fellowship.
Csaba D. Tóth: Supported by the NSF award DMS-2154347.
Tianyi Zhang: Supported by Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118) and the "111 Center" (No. B26023).

1 Introduction

A *shortest-path tree (SPT)* of an undirected edge-weighted n-vertex graph $G = (V, E, \mathbf{w})$ with respect to a designated *source* or *root* vertex $s \in V$, denoted by $\mathsf{SPT}(G, s)$ is a spanning tree T rooted at s that preserves all distances from s, i.e., for every vertex $v \in V$, the distance $d_T(s, v)$ between s and v in T equals their distance $d_G(s, v)$ in G. For a parameter $\alpha \geq 1$, an *α-shallow tree (α-ST)* is a spanning tree T of G of *root-stretch* at most α, i.e., for every $v \in V$, $d_T(s, v) \leq \alpha \cdot d_G(s, v)$. A *minimum spanning tree (MST)* of G, denoted by $\mathsf{MST}(G)$, is a spanning tree T of G of minimum weight. For a parameter $\beta \geq 1$, a *β-light tree (β-LT)* is a spanning tree T of G of *lightness* β, i.e., $\mathbf{w}(T) \leq \beta \cdot \mathbf{w}(\mathsf{MST}(G))$. The SPT and the MST, including their approximate versions, are among the most fundamental graph constructs and have been extensively studied over decades.

A single tree that simultaneously approximates the SPT and the MST is called a *shallow-light tree (SLT)*. For a pair of parameters $\alpha, \beta \geq 1$, a (α, β)-SLT of graph G w.r.t. a designated source $s \in V$ is a spanning tree of G that is both a α-ST and a β-LT. The notion of SLTs was introduced in the pioneering works of Awerbuch et al. [2, 3] and Khuller et al. [18] (see also [8]). They showed that for every $\epsilon > 0$, a $(1 + \epsilon, O(\frac{1}{\epsilon}))$-SLT can be constructed in linear time for every graph G if an SPT and an MST are given. Khuller et al. [18] also showed that this tradeoff is tight, by presenting a *planar* graph for which $\beta = \Omega(\frac{1}{\epsilon})$ for any $(1 + \epsilon, \beta)$-SLT.

The balance between the useful properties of an MST, which provides a light-weight network, and of an SPT, which provides short paths from a designated source to all other vertices, has led to a wide variety of applications across diverse domains. This includes applications in routing [1, 5, 16, 19, 21, 27, 32] and in network and VLSI-circuit design [7, 8, 14, 26], for data gathering and dissemination tasks in overlay networks [17, 20, 30], in the message-passing model of distributed computing [2, 3], and in wireless and sensor networks [31, 4, 9, 23, 22, 28]. Additionally, SLTs are used as building blocks in other related graph structures, such as *light approximate routing trees* [32], *shallow-low-light trees* [11, 12], *light spanners* [3, 25], and others [26, 23, 22]. In particular, in real-world applications, such as VLSI-design and wireless communication networks, the vertices are embedded in Euclidean space, and the edge weights correspond to the metric distances between the nodes.

Low-dimensional Euclidean spaces. Khuller et al. [18] asked whether a better construction of SLTs can be achieved in Euclidean plane, *which is the focus of this work*. Euclidean space $\mathbb{R}^d$, $d \geq 1$, can be modeled as a complete edge-weighted graph $G = (V, E, \mathbf{w})$ induced by a finite set P of points in $\mathbb{R}^d$ with $V = P$, $E = \binom{P}{2}$, and $\mathbf{w} = \|\cdot\|_2$. Elkin and Solomon [13] showed that the upper bound of $(1 + \epsilon, O(\frac{1}{\epsilon}))$-SLTs in general graphs [2, 3, 18] is asymptotically tight even in Euclidean plane: For a set C of $\lceil 1/\epsilon \rceil$ evenly spaced points on a circle, any $(1 + \epsilon, \beta)$-SLT for C for any source $s \in C$ must have $\beta = \Omega(\frac{1}{\epsilon})$. Solomon [29] showed that allowing *Steiner points* lead to substantial improvement in Euclidean plane: For

every set $P \subset \mathbb{R}^2$ and source $s \in P$, one can construct a *Steiner* $(1+\epsilon, O(\sqrt{1/\epsilon}))$-SLT in linear time. Moreover, this bound is asymptotically tight: For the same set C of $\lceil 1/\epsilon \rceil$ evenly spaced points on a circle (in fact, here $\lceil \sqrt{1/\epsilon} \rceil$ evenly spaced points suffice), any Steiner $(1+\epsilon, \beta)$-SLT for C for any source $s \in C$ must have $\beta = \Omega(\sqrt{1/\epsilon})$ [13, 29].

Approximation algorithms and hardness. The aforementioned results provide *tight existential bounds* on the tradeoff between root-stretch and lightness of SLTs in general graphs as well as in planar graphs and in Euclidean plane; moreover, as mentioned above, tight bounds were established also for *Steiner* SLTs in Euclidean plane. However, these algorithms do not necessarily provide instance-optimal SLTs: We present point sets $P \subset \mathbb{R}^2$ for which any previous SLT algorithm in [2, 3, 18] returns a $(1+\epsilon)$-ST of weight $O(\frac{1}{\epsilon}) \cdot \mathbf{w}(\mathsf{MST}(P))$, Solomon [29] constructs a Steiner $(1+\epsilon)$-ST of weight $O(\sqrt{1/\epsilon}) \cdot \mathbf{w}(\mathsf{MST}(P))$, but the minimum weight of a $(1+\epsilon)$-ST is only $O(1) \cdot \mathbf{w}(\mathsf{MST}(P))$. Despite the large body of work on SLTs, very little is known about SLTs from the perspective of optimization and approximation algorithms.

In the *$(1+\epsilon)$-SLT problem*, we are given a parameter $\epsilon \geq 0$ and an edge-weighted graph G, and the goal is to find a $(1+\epsilon)$-ST for G of minimum weight. Khuller et al. [18] showed that for any $\epsilon > 0$, the $(1+\epsilon)$-SLT problem is NP-hard (via a reduction from 3SAT), while the case $\epsilon = 0$ can be solved in near-linear time. Cheong and Lee [6] showed that it is NP-hard in Euclidean plane, as well (via a reduction from Knapsack). A κ-approximation algorithm for the problem should return a $(1+\epsilon)$-ST for G whose weight is at most κ times that of a minimum weight $(1+\epsilon)$-ST. One can also consider a *bicriteria approximation*: a (κ_1, κ_2)-approximation for the problem should return a $(1+\kappa_1 \cdot \epsilon)$-ST for G whose weight is at most κ_2 times that of a minimum-weight $(1+\epsilon)$-ST.

The *tight existential bounds*, mentioned above, yield *approximation factors* of $O(1/\epsilon)$ and $O(\sqrt{1/\epsilon})$, respectively, for the $(1+\epsilon)$-SLT problem on general edge-weighted graphs and in Euclidean plane, respectively. To the best of our knowledge, no other approximation algorithm or hardness result is known for this problem, even for basic graph families such as the complete graph with Euclidean edge weights. (In our full version, we discuss a related problem, for which both approximation algorithms and hardness results are known.)

In Euclidean spaces, one can define the *Steiner $(1+\epsilon)$-SLT problem*: For a parameter $\epsilon \geq 0$ and an input point set $P \subset \mathbb{R}^d$, $d \in \mathbb{N}$, the goal is to find a Steiner $(1+\epsilon)$-ST for P of minimum weight. We note that a minimum weight Steiner $(1+\epsilon)$-ST may be significantly lighter than a minimum weight non-Steiner $(1+\epsilon)$-ST. For example, for a set C of $\lceil 1/\epsilon \rceil$ evenly spaced points on a circle, the ratio between the weights of minimum weight non-Steiner and Steiner $(1+\epsilon)$-STs is $\Theta(\sqrt{1/\epsilon})$.

1.1 Our Contribution

We provide a bicriteria approximation for the $(1+\epsilon)$-SLT problem, where $0 < \epsilon \ll 1$ is an arbitrary parameter, in any constant-dimensional Euclidean space. (We shall assume throughout that ϵ is a sub-constant parameter. If $\epsilon > 0$ is a constant, the algorithm in [18] already provides a constant approximation in linear time.)

▶ **Theorem 1.** *There is an $O(n \log n \cdot \mathrm{polylog}(\epsilon^{-1}))$-time algorithm that, given $\epsilon > 0$, a finite set P of n points in Euclidean plane, including a source $s \in P$, returns a Steiner $(1 + O(\epsilon \log \epsilon^{-1}))$-ST of weight at most $O(\mathsf{opt}_\epsilon \cdot \log \epsilon^{-1})$, where opt_ϵ denotes the minimum weight of a Steiner $(1+\epsilon)$-ST. The result extends without any loss in parameters to Euclidean space $\mathbb{R}^d$, for any constant $d \geq 3$.*

Interestingly, our bicriteria approximation algorithm of Theorem 1 incurs the same $O(\log \epsilon^{-1})$ ratio for both the stretch approximation (to the additive ϵ term) and the weight approximation. With some additional effort and another $\log \epsilon^{-1}$ factor in the weight approximation ratio, our result generalizes to the setting without Steiner points in the plane.

▶ **Theorem 2.** *There is an $O(n \log n \cdot \text{polylog}(\epsilon^{-1}))$-time algorithm that, given $\epsilon > 0$, a finite set P of n points in Euclidean plane, including a source $s \in P$, returns a $(1+O(\epsilon \log \epsilon^{-1}))$-ST of weight at most $O(\mathsf{opt}_\epsilon \cdot \log^2 \epsilon^{-1})$, where opt_ϵ denotes the minimum weight of a $(1+\epsilon)$-ST. The result extends to Euclidean space $\mathbb{R}^d$, for any constant $d \geq 3$, with approximation ratio increasing by a factor of $O(\log \epsilon^{-1})$ and the running time increasing by a factor of $\mathsf{poly}(\epsilon^{-1})$.*

To complement our results, we show that the approximation ratio of our algorithms (with or without using Steiner points) is significantly better than the state-of-the-art algorithms at the instance level. Specifically, we design point sets in Euclidean plane for which any previous algorithm returns a $(1+\epsilon)$-ST of approximation ratio at least $\Omega(\sqrt{1/\epsilon})$ with Steiner points and $\Omega\left(\frac{1}{\epsilon}\right)$ without Steiner points.

▶ **Theorem 3.** *For every $\epsilon > 0$, there exists a set $P \subseteq \mathbb{R}^2$ and a source $s \in P$ such that the minimum weight of a $(1+\epsilon)$-ST (resp., Steiner $(1+\epsilon)$-ST) is $O(1) \cdot \mathbf{w}(\mathsf{MST}(P))$, but any previous algorithm in [2, 3, 18] returns a $(1+\epsilon)$-ST of weight $\Omega(\frac{1}{\epsilon}) \cdot \mathbf{w}(\mathsf{MST}(P))$, and the algorithm in [29] returns a Steiner $(1+\epsilon)$-ST of weight $\Omega(\sqrt{1/\epsilon}) \cdot \mathbf{w}(\mathsf{MST}(P))$.*

To prove Theorem 1 and Theorem 2, we reduced the problem to a set of points, called centered ϵ-net, in a region in a cone with aperture $\sqrt{\epsilon}$, within $\Theta(1)$ distance from the root. The classical lower-bound construction for this problem consists of a set P of uniformly distributed points along a circle of unit radius centered at the root s. However, if P is the subset of points in a cone of angle $\sqrt{\epsilon}$, then there exists a Steiner $(1+\epsilon)$-ST of weight $O(1) \cdot \mathbf{w}(\mathsf{MST}(P))$ [29]. This raises the question: What is the maximum lightness of a Steiner $(1+\epsilon)$-ST for points in a cone of aperture $\sqrt{\epsilon}$? We give a lower bound on the maximum lightness of a minimum-weight Steiner $(1+\epsilon)$-ST for points in a cone of aperture $\sqrt{\epsilon}$.

1.2 Technical Overview

Given a set P of n points in the plane, including a source $s \in P$, and a parameter $\epsilon > 0$, we describe $O(n \log n \cdot \text{polylog}(\epsilon^{-1}))$-time algorithms to construct a (Steiner) $(1+\epsilon \cdot \log \epsilon^{-1})$-ST rooted at s, and then analyze its weight compared to the minimum weight $(1+\epsilon)$-ST rooted at s. We note that, since $(1+\epsilon)$-STs do not have a recursive substructure, the stretch between two arbitrary points in P may be unbounded. Yet, we can apply a divide-and-conquer strategy by clustering nearby points together, based on their position w.r.t. the source s.

In Section 3, we partition the plane into trapezoid tiles, and show that the union of bicriteria approximate SLTs for the point sets in the tiles is a bicriteria approximation for the entire point set for both the Steiner and non-Steiner settings (Theorem 6). We construct a tiling based on geometric considerations. The diameter of each tile τ is proportional to the distance $\text{dist}(s, \tau)$, and the shape of τ is roughly $A \times (A \cdot \sqrt{\epsilon})$ for $A = \text{dist}(s, \tau)$; see Figure 1. That is, we choose the aspect ratio of every tile to be roughly $\sqrt{\epsilon}$ for the following reason: The triangle inequality implies that every ps-path of weight at most $(1+\epsilon) \cdot d(p, s)$ lies in an ellipse $\mathcal{E}_{ps}$ with foci p and s, and aspect ratio roughly $\sqrt{\epsilon}$; see Section 2. Therefore, the union of all ellipses $\mathcal{E}_{ps}$, for all points $p \in P \cap \tau$, will be similar to the tile τ in the sense that the aspect ratio of its bounding box is roughly $\sqrt{\epsilon}$. The shape of the tiles is crucial for the proof of the reduction (Theorem 6).

For all points in a tile $P \cap \tau$, the distance $d(p, s)$ to s is the same up to constant factors. We can further partition the set of points in each tile into cluster by approximating $d(p, s)$ up to a $(1 + \epsilon)$-factor. Recall that a classical ϵ-*net* in a metric space (X, d) is a set $N \subset X$ such that the points in N are at least ϵ distance apart, and the ϵ-neighborhood of every point $x \in X$ contains a net point in N. In Section 3, we define a *centered ϵ-net* N, where points $a, b \in N$ are at least $\epsilon \cdot \max\{d(a, s), d(b, s)\}$ apart, and the $(\epsilon \cdot d(p, s))$-neighborhood of every point $p \in P$ contains a net point in N. We show that a bicriteria approximate STs for a centered ϵ-net can be extended to bicriteria ST for the entire point set, using $O(1)$-spanners in the neighborhoods of the net points (Lemma 7). Interestingly, we reduce the $(1 + \epsilon)$-SLT problem for P to a variant of the Steiner $(1 + \epsilon)$-SLT problem for the net $N \subset P$, where all Steiner points must be in the original set P. We note that although the reduction steps in Section 3 are new and essential to our approach, they are based mainly on standard techniques.

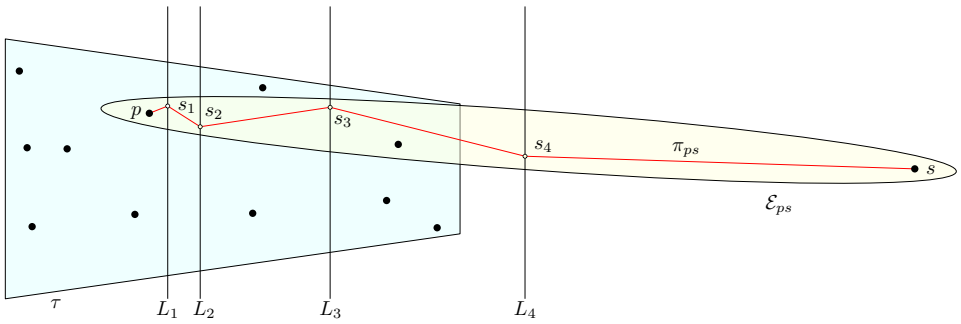


Figure 1 Points in a trapezoid tile τ, and Steiner points $s_1, \ldots, s_4 \in \mathcal{E}_{ps}$ on parallel lines.

The core technical contributions of our work appear in Sections 4 and 5, where we construct a Steiner ST for a centered ϵ-net in Section 4 and then extend the construction to the non-Steiner setting in Section 5. The Steiner setting is easier to work with because we can control the location of Steiner points. We begin with a brief overview of the Steiner construction; refer to Figure 1. From the perspective of a single point $p \in P \cap \tau$, the construction is similar to the Steiner SLT construction by [29], which gave an existentially tight bound: We choose Steiner points $s_1, s_2, \ldots, s_{k-1}$ in the ellipse $\mathcal{E}_{ps}$ on parallel lines $L_1, \ldots, L_{k-1}$ at distance $4^i \epsilon \cdot d(p, s)$ from p, where $k = O(\log \epsilon^{-1})$. Solomon [29] shows that one can carefully choose Steiner points so that the stretch of the path $\pi_{ps} = (p, s_1, \ldots, s_{k-1}, s)$ is at most $1 + \epsilon$. However, geometric calculations show that even if we choose arbitrary points $s_i \in L_i \cap \mathcal{E}_{ps}$ for $i = 1, \ldots, k - 1$, then each edge $s_{i-1}s_i$ still contributes only $O(\epsilon)$ to the stretch (more precisely, $\mathbf{w}(s_{i-1}s_i)$ exceeds the length of its orthogonal projection to the line ps by $O(\epsilon) \cdot d(p, s)$). In other words, arbitrary Steiner points $s_i \in L_i \cap \mathcal{E}_{ps}$, for $i = 1, \ldots, k - 1$, guarantee a root-stretch $1 + O(\epsilon \cdot \log \epsilon^{-1})$.

For a point set $P_\tau = P \cap \tau$ in a tile τ, we follow the above strategy, but we synchronize the lines $L_1, \ldots, L_k$ chosen for different points in P_τ. Then each line L_i corresponds to many points $p \in P_\tau$, and intersects their ellipses $\mathcal{E}_{ps}$. We use a *minimum hitting set* for the intervals $L_i \cap \mathcal{E}_{ps}$, to choose the minimum number of Steiner points that serve all associated ellipses. The weight analysis uses the fact that each ellipse $\mathcal{E}_{ps}$ contains a ps-path that crosses all lines $L_1, \ldots, L_k$.

In Section 5, we adapt the Steiner ST algorithm to the non-Steiner setting. However, both the algorithm design and its analysis are more challenging. Instead of creating Steiner points s_i in a line L_i of our choice, now all points s_i must be in P. We use the lines $L_1, \ldots, L_k$ to

cover the ellipse $\mathcal{E}_{ps}$ with axis-aligned rectangles whose corners are on two consecutive lines L_i and L_{i+1}; and then choose *minimum hitting sets* from P for the *nonempty* rectangles. Some of the rectangles R_i might be *empty* (i.e., $R_i \cap P = \emptyset$). This means that we cannot choose a point $s_i \in P$ in R_i for some ps-path (our algorithm simply skips R_i), but it also means that an optimal ST OPT does not have any vertices in R_i, to the ps-path in OPT contains an edge that traverses R_i whose weight is proportional to the width of R_i. This is a crucial observation for the weight analysis. The root-stretch analysis also requires more work in the non-Steiner setting: In the Steiner case, the Steiner points s_{i-1} and s_i are on the lines L_{i-1} and L_i, so we can control the distance between s_{i-1} and s_i. However, when are limited to points $s_{i-1}, s_i \in P$ in rectangles R_{i-1} and R_i, it is possible that s_{i-1} and s_i are too close to each other, and their contribution to the root-stretch is too large. In such cases, we modify the ps-paths by skipping s_{i-1} or s_i. This ensures that the distances between consecutive points of the ps-paths are sufficiently large, and we prove that the weight increases by at most a constant factor. We show that our algorithms (for both the Steiner and non-Steiner settings) extend naturally to higher dimensions using cone partitioning and approximate high-dimensional hitting sets. For the Steiner version, the approximation guarantee remains unchanged. However, the non-Steiner algorithm incurs an additional $\log(\varepsilon^{-1})$ factor in its approximation ratio. Its running time also increases by an additional $\mathsf{poly}(\varepsilon^{-1})$ factor.

2 Preliminaries

Let $p, s \in \mathbb{R}^2$, and $\epsilon > 0$. If π_{ps} is a ps-path of weight at most $(1+\epsilon) \cdot d(p,s)$, then every point $q \in \pi_{ps}$, we have $d(p,q) + d(q,s) \leq \mathbf{w}(\pi_{ps}) \leq (1+\epsilon) \cdot d(p,s)$. This implies that π_{ps} is contained in the ellipse $\mathcal{E}_{ps}$ with foci p and s, and major axis $(1+\epsilon)d(p,s)$; see Figure 2. The ellipse $\mathcal{E}_{ps}$ is contained in a rectangle spanned by its major and minor axes. The length of its minor axis is $\sqrt{(1+\epsilon)^2 - 1} \cdot d(p,s) = \sqrt{2\epsilon + \epsilon^2} \cdot d(p,s) < 2\sqrt{\epsilon} \cdot d(p,s)$ if $\epsilon < 2$.

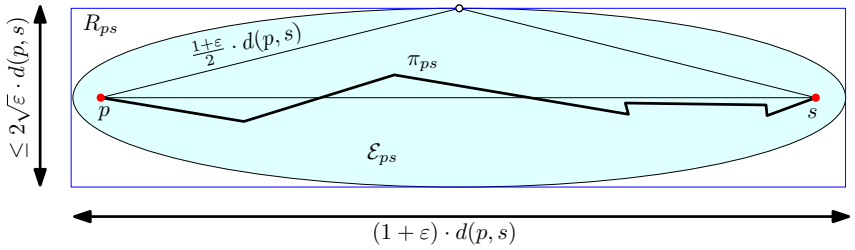


Figure 2 The ellipse $\mathcal{E}_{ps}$ with foci p and s and major axis of length $(1+\epsilon) \cdot d(p,s)$.

For analyzing the weight of an ST, we consider a ps-path π_{ps} as a polyline (i.e., a subset of the plane). In particular, for any region $R \subset \mathbb{R}^2$, the intersection $\pi \cap R$ is the part of the polyline contained in R.

Slopes, slack, and stretch. The *slope* of a line segment ab is $\text{slope}(ab) = \frac{y(b)-y(a)}{x(b)-x(a)}$ if $x(a) \neq x(b)$, and $\text{slope}(ab) = \infty$ if $x(a) = x(b)$. For a line segment ab in the plane, we denote by $\text{proj}(ab)$ the orthogonal projection of ab to the x-axis. Note that $|\text{proj}(ab)| = |x(b) - x(a)|$. We define the *slack* of ab as $\text{slack}(ab) = d(a,b) - |\text{proj}(ab)|$. In our paper, all proofs of theorems, lemmas, corollaries, and observations marked with a $\star$ are in our full version.

▶ **Lemma 4** (⋆). *For any line segment ab with $\text{slope}(ab) \neq \infty$, we have*

$$|\text{proj}(ab)| \cdot \left(1 + \frac{1}{3} \cdot |\text{slope}(ab)|^2\right) \leq d(a,b) \leq |\text{proj}(ab)| \cdot \left(1 + \frac{1}{2} \cdot |\text{slope}(ab)|^2\right) \quad (1)$$

$$\frac{1}{3} \cdot |\text{slope}(ab)|^2 \leq \frac{\text{slack}(ab)}{|\text{proj}(ab)|} \leq \frac{1}{2} \cdot |\text{slope}(ab)|^2. \quad (2)$$

It is known that a x-monotone path (i.e., a path in which the x-coordinates of the points along the path are monotone increasing) with edges of bounded slopes have small stretch.

▶ **Lemma 5.** *Let $\pi = (v_0, v_1, \ldots, v_k)$ be a x-monotone polygonal path in $\mathbb{R}^2$ such that $|\text{slope}(v_{i-1}v_i)| \leq \varrho$ for all $i \in [1,k]$. Then $|\text{slope}(v_0v_k)| \leq \varrho$; and $\frac{\mathbf{w}(\pi)}{d(v_0,v_k)} \leq \sqrt{1+\varrho^2} < 1 + \frac{\varrho^2}{2}$.*

For two points $a, b \in \mathbb{R}^2$ and a parameter $\epsilon > 0$, let $\mathcal{E}_{ab,\epsilon}$ denote the ellipse with foci a and b and major axis $(1+\epsilon) \cdot d(a,b)$, that is, $\mathcal{E}_{ab,\epsilon} = \{p \in \mathbb{R}^2 : d(a,p) + d(p,b) \leq (1+\epsilon) \cdot d(a,b)\}$.

3 Reduction to Net Points in a Trapezoid

In this section, we reduce the problem of constructing a bicriteria approximation for the minimum weight (Steiner) $(1+\epsilon)$-ST to the special case where all points, except the source s, lie in a trapezoid, and the point set is sparse (i.e., form a centered ϵ-net, defined below).

Given a source $s \in \mathbb{R}^2$ and $\epsilon > 0$, define a *tiling* of the plane into a set $\mathcal{T}$ of trapezoids as follows; refer to Figure 3. Let C be a circle of unit radius centered at s. Let O_1 be a regular k-polygon with inscribed circle C, where k is the minimum integer such that the side length of O_1 is less than $\sqrt{\epsilon}$. For all integers $i \in \mathbb{Z}$, let $O_i = 2^i \cdot O_1$, that is, a scaled copy of O_1, centered at s. Finally, add k rays emanating from s that pass through the vertices of the polygons O_i, $i \in \mathbb{Z}$. Polygons O_i, $i \in \mathbb{Z}$, and the k rays subdivide the plane into a set $\mathcal{T}$ of trapezoids: Each trapezoid $\tau \in \mathcal{T}$ lies between two consecutive polygons O_i and O_{i+1}, and two consecutive rays.

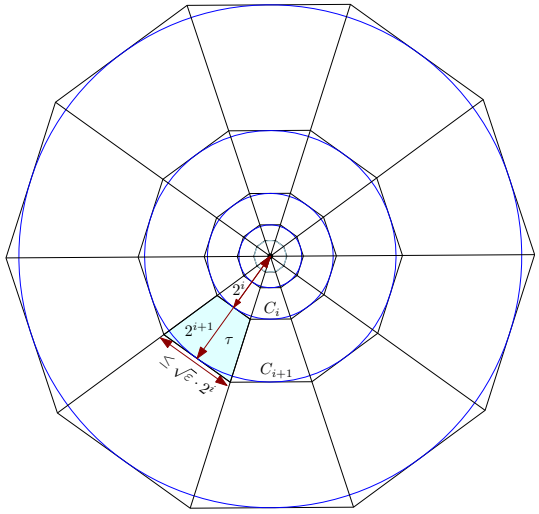


Figure 3 A section of the trapezoid tiling of the plane.

Let P be a set of n points in the plane, and let $\tau_1, \ldots, \tau_m \in \mathcal{T}$ be a finite set of tiles that cover P, and let $P_i := P \cap \tau_i$. For a tile $\tau_i \in \mathcal{T}$, we define a *tile-restricted* $(1+\epsilon)$-*ST* (for short, $(1+\epsilon)$-*tST*) as a tree $T = (V, E)$ such that $P_i \cup \{s\} \subseteq V(T) \subseteq P$, and T contains a ps-path of weight at most $(1+\epsilon) \cdot d(p,s)$ for every $p \in P_i$. In other words, a tile-restricted $(1+\epsilon)$-*ST* is a Steiner $(1+\epsilon)$-ST for $P_i \cup \{s\}$ rooted at s, where all Steiner points are in P.

▶ **Theorem 6** (⋆). *Given a set $P \subset \mathbb{R}^2$ of n points, a source $s \in \mathbb{R}^2$, a parameter $\epsilon > 0$, and two real functions $f(.)$ and $g(.)$. Let $\tau_1, \ldots, \tau_m$ be the set of tiles in $\mathcal{T}$, where $P_i := P \cap \tau_i \neq \emptyset$. Let G_i be a tile-restricted $(1 + f(\epsilon) \cdot \epsilon)$-ST (resp., Steiner $(1 + f(\epsilon) \cdot \epsilon)$-ST) for $P_i \cup \{s\}$ of weight $\mathbf{w}(G_i) \leq O(g(\epsilon) \cdot \mathsf{opt}_{\epsilon,i})$, where $\mathsf{opt}_{\epsilon,i}$ is the minimum weight of a $(1+\epsilon)$-ST for $P_i \cup \{s\}$, for all $i = 1, \ldots, m$. Then, a shortest-path tree of the graph $G := \bigcup_{i=1}^{m} G_i$, rooted at s, is a $(1+f(\epsilon)\cdot\epsilon)$-ST (resp., Steiner $(1+f(\epsilon)\cdot\epsilon)$-ST) for $P \cup \{s\}$ and $\mathbf{w}(G) \leq O(g(\epsilon)\cdot \mathsf{opt}_\epsilon)$, where opt_ϵ is the minimum weight of a $(1+\epsilon)$-ST for $P \cup \{s\}$.*

For the purpose of a $(1+\epsilon)$-ST, with a source (or center) $s \in P$, we define a variation of ϵ-nets where the density of the net depends on the distance from s: A *centered ϵ-net* (for short, *ϵ-cnet*) with center s is a subset $N \subset P$ such that

- for all $a, b \in N$, $a \neq b$, we have $d(a,b) > \epsilon \cdot \min\{d(a,s), d(b,s)\}$ and
- for every $p \in P$, there exists a point $a \in N$ such that $d(p,a) \leq \epsilon \cdot d(p,s)$.

Given a point set $P \subset \mathbb{R}^2$, including $s \in P$, a parameter $\epsilon > 0$ and a subset P, we also define a *net-restricted $(1+\epsilon)$-ST for $N \subset P$* as a tree $T = (V, E)$ such that $N \cup \{s\} \subseteq V(T) \subseteq P$, and T contains a ps-path of weight at most $(1+\epsilon) \cdot d(p,s)$ for every $p \in N$. This means that T is a Steiner $(1+\epsilon)$-ST rooted at s for the ϵ-cnet N, where all Steiner points are in P. We claim that it suffices to find a net-restricted (resp., Steiner) $(1+\epsilon)$-ST for a ϵ-cnet with center s.

▶ **Lemma 7** (⋆). *Let P be a set of n points in the plane, $s \in P$, $\epsilon \in (0, \frac{1}{9})$, and $N \subset P$ a ϵ-cnet for P. Let $f(.)$ and $g(.)$ be real functions such that $f(x) \geq x$ and $g(x) \geq x$ for all $x \geq 0$. Assume that in $O(|N| \log |N|)$ time, we can compute a net-restricted (resp., Steiner) $(1+f(\epsilon)\cdot\epsilon)$-ST T_N for N of weight $O(g(\epsilon)) \cdot \mathsf{opt}_\epsilon(N)$, where $\mathsf{opt}_\epsilon(N)$ is the minimum weight of a net-restricted (resp., Steiner) $(1+\epsilon)$-ST for N. Then, in $O(n \log n)$ time, we can compute a (Steiner) $(1+O(f(\epsilon)\cdot\epsilon))$-ST T_P for P of weight $O(g(\epsilon)) \cdot \mathsf{opt}_\epsilon(P)$, where $\mathsf{opt}_\epsilon(P)$ is the minimum weight of a (Steiner) $(1+\epsilon)$-ST for P.*

4 Steiner Shallow Trees for Points in a Tile

In this section, we prove Theorem 1. By Theorem 6 and Lemma 7, we may assume that P is a set of points in a trapezoid $\tau \in \mathcal{T}$ (defined in Section 3), and P is a centered ϵ-net w.r.t. center $s \in P$. We may further assume w.l.o.g. that $\tau \subset [0,1] \times [-\sqrt{\epsilon}, \sqrt{\epsilon}]$ and $s = (2,0)$; see Figure 4. We first present an algorithm that constructs a Steiner tree T for $P \cup \{s\}$, rooted at s. Then we show that T is a $(1+O(\epsilon \log \epsilon^{-1}))$-ST and its weight is $O(\mathsf{opt}_\epsilon \cdot \log \epsilon^{-1})$, where opt_ϵ denotes the minimum weight of a Steiner $(1+\epsilon)$-ST for P rooted at s; and let OPT_ϵ be one such Steiner $(1+\epsilon)$-ST.

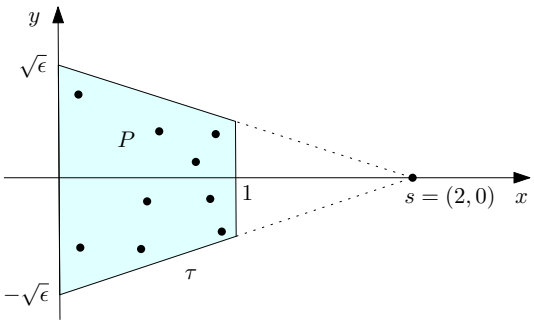


Figure 4 A point set P in a trapezoid τ and a source $s = (2,0)$.

Steiner shallow tree construction: Overview. We start with a brief overview of the ST construction and then present the algorithm in detail. We construct a Steiner graph G on $P \cup \{s\}$ (which is not necessarily a tree), and then let T be the shortest path tree of G rooted at s (that is, the single-source shortest path tree of G with s as the source). We can analyze the root-stretch in the graph G, and the weight of T is upper-bounded by the weight of G.

For each point $p \in P$, we know that OPT_ϵ contains a ps-path of length at most $(1+\epsilon)\cdot d(p,s)$ in the ellipse $\mathcal{E}_{ps}$ with major axis $(1+\epsilon)\cdot d(p,s)$. We associate each point $p \in P$ with vertical lines $L_0(p), \ldots, L_{k-1}(p)$, where $k = O(\log \epsilon^{-1})$. For $i = 0, 1, \ldots, k-1$, we choose a Steiner point s_i in the line segments $\mathcal{E}_{ps} \cap L_i(p)$, and add the path $\pi_{ps} = (p, s_0, \ldots, s_{k-1}, s)$ to the graph G. We show (Lemma 12) that if the distances between the vertical lines $L_1, \ldots, L_k$ increase exponentially, then $\mathbf{w}(\pi_{ps}) = (1 + O(\epsilon \log \epsilon^{-1})) \cdot d(p,s)$.

Importantly, each line L_i is associated with multiple points in P, and each Steiner point $s_i \in L_i$ also serves multiple points in P. For each line L_i, we choose the Steiner points in L_i as a minimum hitting set for the line segments $\mathcal{E}_{ps} \cap L_i$ over all points $p \in P$ associated with line L_i. Since the intersection $\mathsf{OPT}_\epsilon \cap L_i$ is a hitting set for these intervals, we can charge the weight of G to the weight of OPT_ϵ. The approximation factor $O(\log \epsilon^{-1})$, in the stretch and the weight, is the result of using $O(\log \epsilon^{-1})$ Steiner points, one in each line L_i, for $i = 1, \ldots, k$.

Steiner shallow tree construction: Details. Assume w.l.o.g. that $\epsilon = 4^{-(k+1)}$ for some $k \in \mathbb{N}$. Each ellipse $\mathcal{E}_{ps}$ is contained in an ellipse $\mathcal{E}_{pb(p),2\epsilon}$ ($\star$), where the segment $pb(p)$ is horizontal and $d(p,b(p)) \le 2 \cdot d(p,s)$. As a shorthand notation, we use $\mathcal{E}_p := \mathcal{E}_{pb(p),2\epsilon}$.

We define families of vertical lines. For every integer $i \ge 0$, let

$$\mathcal{L}_i = \{x = j4^i \cdot \epsilon : j \in \mathbb{Z}\}, \tag{3}$$

that is, the distance between two consecutive vertical lines in $\mathcal{L}_i$ is $4^i \cdot \epsilon$. For every point $p \in P$, we recursively choose $L_i(p) \in \mathcal{L}_i$ for $i = 0, 1, \ldots, k-1$, as follows. Let $L_0(p)$ be the second line in $\mathcal{L}_0$ to the right of p. For $i = 1, \ldots, k-1$, if $L_{i-1}(p)$ has already been chosen, let $L_i(p)$ be the second line in $\mathcal{L}_i$ to the right of $L_{i-1}(p)$. (We choose the second line in $\mathcal{L}_i$, rather than the first one, to ensure that the gaps between $L_{i-1}(p)$ and $L_i(p)$ grow exponentially in i; cf. Observation 8.)

Now consider a line $L \in \mathcal{L}_i$, and let $P(L)$ be the set of all points in P associated with L. Formally, we put $P(L) = \{p \in P : L_i(p) = L\}$. Consider the set of intervals $I(L) = \{L \cap \mathcal{E}_p : p \in P(L)\}$. Let $H(L)$ be a *minimum hitting set* (a.k.a., piercing set, stabbing set, or transversal) for $I(L)$: It is a minimum subset of L that contains at least one point in each interval in $I(L)$; it can be computed in $O(|I(L)| \log |H(L)|)$ time [10, 15, 24].

Finally, we construct the Steiner graph G as follows. The vertex set of G comprises P, s, and the points $H(L)$ for all $L \in \mathcal{L}_i$, $i = 0, \ldots, k-1$. The edges are defined as follows. For each point $p \in P$, consider the lines $L_0(p), \ldots, L_{k-1}(p)$ associated with p. For $i = 0, 1, \ldots, k-1$ let $s_i(p)$ be an arbitrary point in $H(L_i(p)) \cap \mathcal{E}_p$. Add the edges of the path $\pi_{ps} = (p, s_0(p), \ldots, s_{k-1}(p), s)$ to G. We output the shortest path tree T of G rooted at s.

▶ **Observation 8.** *For $i \in [1, k-1]$, $\mathsf{dist}(L_{i-1}(p), L_i(p))$ is within $[4^i\,\epsilon, 2 \cdot 4^i\,\epsilon]$.*

From Observation 8, using a geometric series, we obtain the following corollary:

▶ **Corollary 9** ($\star$). *For $i \in [1, k-1]$, $\mathsf{dist}(p, L_i(p))$ is within $[4^i\,\epsilon, \frac{32}{3} \cdot 4^i\,\epsilon]$.*

Root stretch analysis. It is enough to analyze the root stretch in the graph G. Recall that in the construction of G, we have already built a path $\pi_{ps} = (p, s_0(p), \ldots, s_{k-1}(p), s)$. We show that $\mathbf{w}(\pi_{ps}) \leq (1 + O(\epsilon \log \epsilon^{-1})) \cdot d(p, s)$ (Lemma 12). We first need to estimate the slopes of the edges of π_{ps}. We start with the following lemma.

▶ **Lemma 10.** *For every point $p \in P$ if a vertical line L is to the right of p at distance $x \in [0, \frac{2}{3}]$ from p, then $\mathbf{w}(L \cap \mathcal{E}_p) = \Theta(\sqrt{\epsilon\, x})$. In particular, for every $i = 0, \ldots, k-1$, we have $\mathbf{w}(L_i(p) \cap \mathcal{E}_p) = \Theta(2^i \cdot \epsilon)$.*

Proof. Consider the unit disk $D = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 = 1\}$ and the line $L : x = -1 + \varrho$ for $0 \leq \varrho \leq 1$; see Figure 5. Then for $\varrho \in [0, 1]$, we obtain

$$\mathbf{w}(D \cap L) = 2 \cdot \sqrt{1^2 - (1-\varrho)^2} = 2 \cdot \sqrt{2\varrho - \varrho^2} = 2^{3/2} \cdot \sqrt{\varrho - \frac{\varrho^2}{2}} = \Theta(\sqrt{\varrho}).$$

The ellipse $\mathcal{E}_p = \mathcal{E}_{pb,2\epsilon}$ is an affine image of the unit disk D: The affine transformation $f(x, y) = (\frac{M}{2}x, \frac{m}{2}y)$ takes D to $\mathcal{E}_p$, where M and m are the major and minor axes of $\mathcal{E}_p$. We estimate M and m up to constant factors. Since $P \subset [0, 1] \times [-\sqrt{\epsilon}, \sqrt{\epsilon}]$ and $s = (2, 0)$, then we have $1 \leq |x(p) - x(s)| \leq 2$. We have $1 \leq d(p, b(p)) \leq 4$ $(\star)$. The major axis of $\mathcal{E}_p$ is $M = (1 + 2\epsilon) \cdot d(p, b(p))$ so $1 < M \leq 4 + 8\epsilon$; and its minor axis is $m = 2 \cdot \sqrt{(1 + 2\epsilon)^2 - 1} \cdot d(p, b(p)) = 4 \cdot \sqrt{\epsilon + \epsilon^2} \cdot d(p, b(p))$, and so $4 \cdot \sqrt{\epsilon} \leq m \leq 16 \cdot \sqrt{\epsilon + \epsilon^2} \leq 16\sqrt{2} \cdot \sqrt{\epsilon}$.

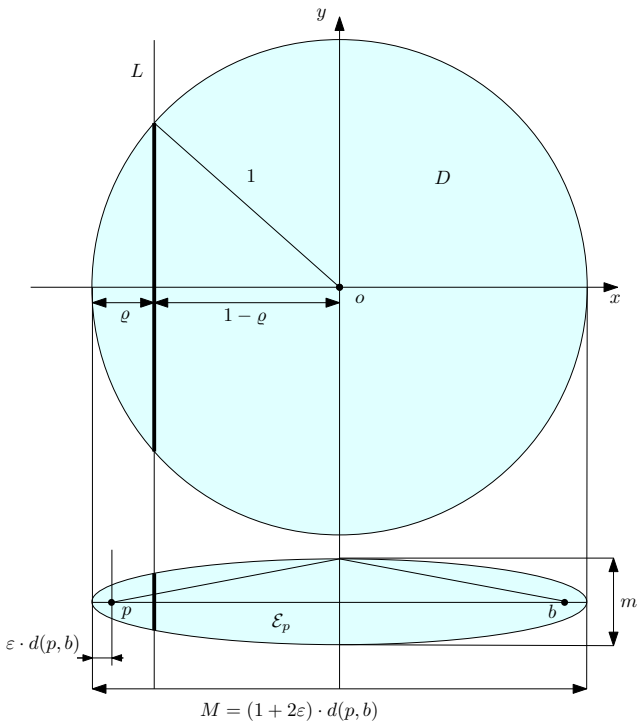


Figure 5 A unit disk D, a cross section $L \cap D$, and an affine transformation to the ellipse $\mathcal{E}_p$.

The focus p is at distance $\epsilon \cdot d(p, b(p))$ from the leftmost point of $\mathcal{E}_p$, where $\epsilon \leq \epsilon \cdot d(p, b(p)) \leq 4\epsilon$. By Corollary 9, the distance between $L_i(p)$ and the leftmost point of $\mathcal{E}_p$ is at least $\epsilon + 4^i\epsilon = \Omega(4^i\epsilon \cdot M)$ and at most $4\epsilon + \frac{32}{3} \cdot 4^i\epsilon = O(4^i\epsilon \cdot M)$. Overall, this distance is $\Theta(4^i\epsilon \cdot M)$.

The inverse transformation f^{-1} takes $\mathcal{E}_p$ to the unit disk D, and the line $L_i(p)$ to a vertical line $L : x = -1 + \varrho$, where $\varrho = \Theta(4^i\epsilon)$. As noted above, we have $\mathbf{w}(L \cap D) = \Theta(\sqrt{\varrho}) = \Theta(2^i \cdot \sqrt{\epsilon})$, consequently $\mathbf{w}(L_i(p) \cap \mathcal{E}_p) = \Theta(m \cdot \sqrt{\varrho}) = \Theta(\sqrt{\epsilon} \cdot 2^i \cdot \sqrt{\epsilon}) = \Theta(2^i \cdot \epsilon)$. ◀

Combining Observation 8 and Lemma 10, we have the following corollary:

▶ **Corollary 11** (⋆). *For every $p \in P$, consider the path $\pi_{ps} = (p, s_0, \ldots, s_{k-1}, s)$. For all $i = 1, \ldots, k-1$, we have $|\text{slope}(s_{i-1}s_i)| \le O(2^{-i})$.*

▶ **Lemma 12.** *For every $p \in P$, we have $\mathbf{w}(\pi_{ps}) \le (1 + O(\epsilon \log \epsilon^{-1})) \cdot d(p, s)$.*

Proof. By construction, we have $\pi_{ps} = (p, s_0, s_1, \ldots s_{k-1}, s)$. We partition π_{ps} into three parts: ps_0, $(s_0, s_1, \ldots s_{k-1})$ and $s_{k-1}s$, and bound the weight of each part separately.

First we estimate the weight of the first edge ps_0. Recall that $p \in [0,1] \times [-\sqrt{\epsilon}, \sqrt{\epsilon}]$ and $s = (2, 0)$. By construction, we have $|x(p) - x(s_0)| \le \epsilon$, and Lemma 10 gives $|y(p) - y(s_0)| \le O(\epsilon)$. Therefore, we have $\mathbf{w}(ps_0) = O(\epsilon) \le O(\epsilon) \cdot d(p, s)$. We can now bound the weight of the subpath $(s_0, s_1, \ldots, s_{k-1})$ of π_{ps}. By Lemma 4, Observation 8, and Corollary 11,

$$\begin{aligned}
\mathbf{w}((s_0, s_1, \ldots, s_{k-1})) &= \sum_{j=1}^{k} \mathbf{w}(s_{j-1}s_j) \le \sum_{j=1}^{k-1} \left(1 + \frac{(\text{slope}(s_{j-1}s_j))^2}{2}\right) \cdot |\text{proj}(s_{j-1}s_j)| \\
&= \sum_{j=1}^{k-1} |\text{proj}(s_{j-1}s_j)| + \sum_{j=1}^{k-1} O(4^{-j}) \cdot |\text{proj}(s_{j-1}s_j)| \\
&\le |\text{proj}(s_0 s_{k-1})| + \sum_{j=1}^{k-1} O(4^{-j}) \cdot 2 \cdot 4^j \cdot \epsilon = |\text{proj}(s_0 s_{k-1})| + O(k\epsilon) \\
&= |\text{proj}(s_0 s_{k-1})| + O\left(\epsilon \log \epsilon^{-1}\right) = |\text{proj}(s_0 s_{k-1})| \left(1 + O\left(\epsilon \log \epsilon^{-1}\right)\right).
\end{aligned}$$

Finally, we estimate the weight of the last edge, $s_{k-1}s$, of π_{ps}. We have

$$\text{dist}(p, L_{k-1}(p)) < \frac{32}{3} \cdot 4^{k-1} \cdot \epsilon = \frac{2}{3\epsilon} \cdot \epsilon = \frac{2}{3}. \qquad \text{(by Corollary 9 and } \epsilon = 4^{-(k+1)}) \qquad (4)$$

Since $x(p) \in [0, 1]$ and $s = (2, 0)$, then $|x(s_{k-1}) - x(s)| \ge \frac{1}{3}$. Lemma 10 gives $|y(s_{k-1}) - y(s)| = |y(s_{k-1})| \le O(2^{k-1} \cdot \epsilon) = O\left(\frac{1}{\sqrt{\epsilon}} \cdot \epsilon\right) = O(\sqrt{\epsilon})$. Consequently, $\text{slope}(s_{k-1}s) = O(\sqrt{\epsilon})$, and Lemma 4 yields $\mathbf{w}(s_{k-1}s) \le (1 + O(\epsilon)) \cdot |\text{proj}(s_{k-1}s)|$.

The sum of the weights of the three parts is

$$\begin{aligned}
\mathbf{w}(\pi_{ps}) &= \mathbf{w}(ps) + \mathbf{w}((s_0, s_1, \ldots, s_{k-1})) + \mathbf{w}(s_{k-1}s) \\
&\le O(\epsilon) \cdot d(p, s) + \left(1 + O\left(\epsilon \log \epsilon^{-1}\right)\right) |\text{proj}(s_0 s_{k-1})| + (1 + O(\epsilon)) \cdot |\text{proj}(s_{k-1}s)| \\
&\le O(\epsilon) \cdot d(p, s) + \left(1 + O\left(\epsilon \log \epsilon^{-1}\right)\right) |\text{proj}(s_0 s)| \le \left(1 + O\left(\epsilon \log \epsilon^{-1}\right)\right) d(p, s).
\end{aligned}$$

◀

Weight analysis. We give an upper bound for $\mathbf{w}(G)$. Similarly to the root stretch analysis, we decompose π_{ps} into three parts, ps_0, $(s_0, s_1, \ldots, s_{k-1})$, and $s_{k-1}s$, and then bound the weight of the union of each.

We first analyze the total weight of the union of edges $s_{i-1}(p)\, s_i(p)$ of the paths π_{ps} over all $p \in P$. For lines $L_a \in \mathcal{L}_{i-1}$ and $L_b \in \mathcal{L}_i$, let $G(L_a, L_b)$ denote the set of all edges $uv \in E(G)$ such that $u \in I(L_a)$ and $v \in I(L_b)$. Recall that OPT_ϵ is a minimum-weight Steiner $(1+\epsilon)$-ST for $P \cup \{s\}$ rooted at s ($\mathbf{w}(\mathsf{OPT}_\epsilon) = \mathsf{opt}_\epsilon$). We denote by $\mathsf{OPT}_\epsilon \cap [\![L_a, L_b]\!]$ the part of OPT_ϵ clipped in the vertical strip $[\![L_a, L_b]\!]$. Our main lemma is as follows.

▶ **Lemma 13.** *For every $p \in P$ and $i \in \{1, \dots, k-1\}$, we have*

$$\mathbf{w}(G(L_{i-1}(p), L_i(p))) \leq O(\mathbf{w}(\mathsf{OPT}_\epsilon \cap [\![L_{i-1}(p), L_i(p)]\!])).$$

Before the proof of Lemma 13, we observe that it implies the bound on the total weight of the interior edges of the paths π_{ps} over all $p \in P$.

▶ **Corollary 14** (⋆). $\mathbf{w}\left(\bigcup_{p \in P}(s_0(p), \dots, s_{k-1}(p))\right) \leq O(\log \epsilon^{-1} \cdot \mathsf{opt}_\epsilon)$.

It remains to prove Lemma 13. We do this in a sequence of lemmas, using geometric properties of ellipses and interval graphs. We start with an easy observation.

▶ **Observation 15** (⋆). *For every $p \in P$ and every $i \in \{1, \dots, k-1\}$, we have*

$$\mathbf{w}(\mathsf{OPT}_\epsilon \cap [\![L_{i-1}(p), L_i(p)]\!] \cap \mathcal{E}_p) \geq \mathrm{dist}(L_{i-1}(p), L_i(p)).$$

▶ **Lemma 16.** *For every $i \in \{1, \dots, k-1\}$ and every $p \in P$, there exists a set $Q \subset P(L_{i-1}(p))$ of size $|Q| \geq \Omega(|H(L_i(p))|)$ such that the regions in $\{\mathcal{E}_q \cap [\![L_{i-1}(p), L_i(p)]\!] : q \in Q\}$ are disjoint.*

Proof. Each interval in $I(L_i(p))$ has weight $\Theta(2^i \cdot \epsilon)$ by Lemma 10. Let $c \geq 1$ be the ratio of the maximum to the minimum weight of an interval in $I(L_i(p))$. Each interval in $I(L_i(p))$ is of the form $L_i(p) \cap \mathcal{E}_r$ for some point $r \in P(L_i(p))$. For every $r \in P(L_i(p))$, let B_r be the axis-aligned bounding box of $\mathcal{E}_r \cap [\![L_{i-1}(p), L_i(p)]\!]$; see Figure 6. Note that the major axis of $\mathcal{E}_r$ is horizontal, and its minor axis is to the right of L_i by Equation (4). Consequently, for every vertical line $L \subset [\![L_{i-1}(p), L_i(p)]\!]$, we have $\mathbf{w}(L \cap \mathcal{E}_r) \leq \mathbf{w}(L_i(p) \cap \mathcal{E}_r)$, which holds with equality for $L = L_i(p)$. In particular, the height of B_r is $\mathbf{w}(L_i(p) \cap \mathcal{E}_r) = \Theta(2^i \cdot \epsilon)$.

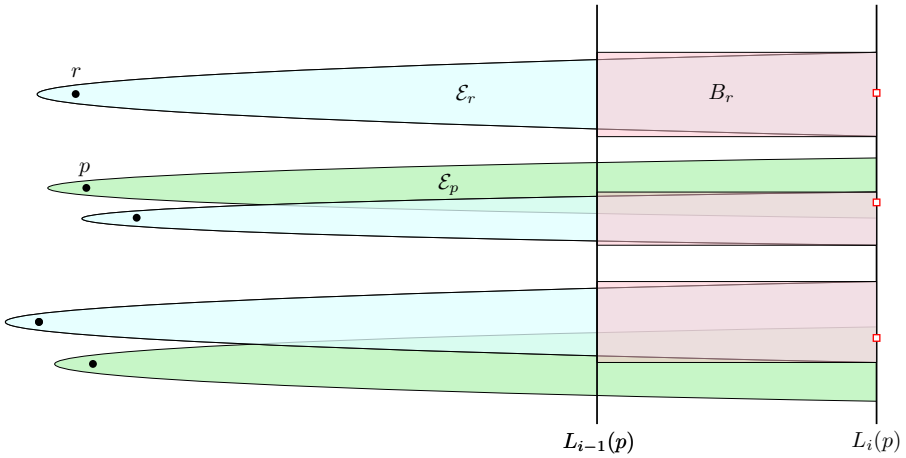


Figure 6 A point $p \in P$ and ellipses $\mathcal{E}_r$ for five points in $P(L_{i-1}(p))$. The boxes B_r corresponding to the three light blue ellipses are disjoint. A minimum hitting set $H(L_i(p))$ has size 3.

Now consider a maximum set $Q_0 \subset P(L_i(p))$ such that the intervals $\{L_i(p) \cap \mathcal{E}_q : q \in Q_0\}$ are disjoint. Note that $|Q_0| = |H(L_i(p))|$, that is, the maximum independent set has the same size as a hitting set for intervals in a line.

A set of disjoint intervals along the line $L_i(p)$ have a well-defined total order, which defines a total order on Q_0. Let $Q \subset Q_0$ be the subset that corresponds to the first and every $\lceil c \rceil$-th interval. Then $|Q_0| = \Theta(|Q|) = \Theta(|H(L_i(p))|)$. Furthermore, the intervals $\{L_i(p) \cap \mathcal{E}_q : q \in Q\}$ are disjoint. This, in turn, implies that the boxes $\{B_q : q \in Q\}$ are disjoint. Thus, the regions $\{\mathcal{E}_q \cap [\![L_{i-1}(p), L_i(p)]\!] : q \in Q\}$ are also disjoint. ◀

Observation 15 and Lemma 16 imply the following corollary.

▶ **Corollary 17** (⋆). *For every $p \in P$ and $i \in \{1, \dots, k-1\}$, we have*

$$\Omega(|H(L_i(p))|) \cdot \mathrm{dist}(L_{i-1}(p), L_i(p)) \leq \mathbf{w}\big(\mathsf{OPT}_\epsilon \cap [\![L_{i-1}(p), L_i(p)]\!]\big).$$

▶ **Lemma 18.** *For every $p \in P$ and $i \in \{1, \dots, k-1\}$, we have*

$$\mathbf{w}(G(L_{i-1}(p), L_i(p))) \leq O(|H(L_i(p)|) \cdot \mathrm{dist}(L_{i-1}(p), L_i(p)).$$

Proof. We first show that every edge in $G(L_{i-1}(p), L_i(p))$ has weight $\Theta(\mathrm{dist}(L_{i-1}(p), L_i(p)))$. Every edge of $G(L_{i-1}(p), L_i(p))$ is the edge $s_{i-1}(r)s_i(r)$ of π_{rs} for some point $r \in P(L_{i-1}(p))$. By Corollary 11, $|\mathrm{slope}(s_{i-1}(r)s_i(r))| \leq O(2^{-i})$, where $i = O(\log_4 \epsilon^{-1})$. Consequently, we have $|\mathrm{slope}(s_{i-1}(r)s_i(r))| \leq O(1)$ for all $i = 1, \dots, k-1$. Therefore,

$$\mathbf{w}(s_{i-1}(r)s_i(r)) \leq O(1) \cdot |\mathrm{proj}(s_{i-1}(r)s_i(r))| \leq O(\mathrm{dist}(L_{i-1}(p), L_i(p))).$$

Next, we show that every vertex $v \in H(L_i(p))$ of $G(L_{i-1}(p), L_i(p))$ has $O(1)$ degree. Lemma 10 gives $\mathbf{w}(L_{i-1}(p) \cap \mathcal{E}_r) = \Theta(2^i)$ and $\mathbf{w}(L_i(p) \cap \mathcal{E}_r) = \Theta(2^{i-1})$ for all $r \in P(L_{i-1}(p))$. Note also that both $L_{i-1}(p) \cap \mathcal{E}_r$ and $L_i(p) \cap \mathcal{E}_r$ have a reflection symmetry in the horizontal major axis of $\mathcal{E}_r$, implying $\bigcup\{L_i(p) \cap \mathcal{E}_r : r \in P(L_i(p)) \text{ and } v \in \mathcal{E}_r\}$ is contained in an interval of length $2 \cdot \Theta(2^i) = \Theta(2^i)$ centered at v. Similarly, $\bigcup\{L_{i-1}(p) \cap \mathcal{E}_r : r \in P(L_{i-1}(p)) \text{ and } v \in \mathcal{E}_r\}$ is contained in an interval $I \subset L_{i-1}$ of length $2 \cdot \Theta(2^{i-1}) = \Theta(2^i)$. Since $\mathbf{w}(L_{i-1}(p) \cap \mathcal{E}_r) = \Theta(2^i)$ for all $r \in P(L_{i-1}(p))$, then the hitting set $H(L_{i-1}(p))$ contains $O(1)$ points in the interval I. In the graph $G(L_{i-1}(p), L_i(p))$, all neighbors of vertex v are in the interval I and in $H(L_{i-1}(p))$. This proves that v has $O(1)$ neighbors in $G(L_{i-1}(p), L_i(p))$.

Overall, every edge of $G(L_{i-1}(p), L_i(p))$ has weight $\Theta(\mathrm{dist}(L_{i-1}(p), L_i(p)))$; and every vertex $v \in H(L_i(p))$ of $G(L_{i-1}(p), L_i(p))$ has $O(1)$ degree. Since $G(L_{i-1}(p), L_i(p))$ is a bipartite graph with partite sets $H(L_{i-1}(p))$ and $H(L_i(p))$, then the total weight of $G(L_{i-1}(p), L_i(p))$ is $O(|H(L_i(p)|) \cdot \mathrm{dist}(L_{i-1}(p), L_i(p))$, as claimed. ◀

Combining Lemma 18 and Corollary 17, we obtain Lemma 13. We conclude:

▶ **Lemma 19.** *For a ϵ-cnet $P \subset [0,1] \times [-\sqrt{\epsilon}, \sqrt{\epsilon}]$ and $s = (0,2)$, the weight of G is $O(\log \epsilon^{-1}) \cdot \mathsf{opt}_\epsilon$, where opt_ϵ denotes the minimum weight of a Steiner $(1+\epsilon)$-ST for P.*

Proof. We have shown that for every $p \in P$, the first edge of π_{ps} has weight $O(\epsilon)$. Consequently, the total weight of the union of first segments of π_{ps} over all $p \in P$ is $O(\epsilon \cdot |P|)$. Since P is a ϵ-cnet, then this is bounded by $O(\mathsf{opt}_\epsilon)$. By Corollary 14, the total weight of the interior edges of the paths π_{ps} over all $p \in P$, namely, $\mathbf{w}\left(\bigcup_{p \in P}(s_0(p), \dots, s_{k-1}(p))\right)$ is bounded by $O(\log \epsilon^{-1}) \cdot \mathsf{opt}_\epsilon$.

Finally, we bound the total weight of the last edges of the paths $\pi_{ps} = (p, s_0(p), \dots, s_{k-1}(p), s)$ for all $p \in P$. By Observation 8, there are $O(1)$ vertical lines in $\mathcal{L}_{k-1}$ between $o = (0,0)$ and $s = (2,0)$. By Corollary 11, we have $\mathbf{w}(\mathcal{E}_p \cap L_{k-1}(p)) = \Theta(\sqrt{\epsilon})$. Consequently, each piercing set $H(L_{k-1}(p))$ has $O(1)$ size. Overall, there are $O(1)$ distinct segments in $\{s_{k-1}(p)s : p \in P\}$, and the weight of each segment is $O(1)$. Consequently, the total weight of these segments is also $O(1)$. Since $\mathrm{dist}(P, s) \geq 1$, then $\mathbf{w}(OPT_\epsilon) \geq 1$, and so $\mathbf{w}(\{s_{k-1}(p)s : p \in P\}) \leq O(1) \leq O(\mathbf{w}(\mathsf{OPT}_\epsilon))$. ◀

Running time. The running time is $O(n \log n \cdot \mathrm{polylog}(\epsilon^{-1}))$.

5 Shallow Trees for Points in a Tile

In this section, we prove Theorem 2. By Theorem 6 and Lemma 7, we may reduce to the problem of finding a tile-restricted $(1+\epsilon)$-ST for points in a tile τ. Let P be a set of n points in the plane, $\tau \in \mathcal{T}$ be a tile, and N_τ an ϵ-cnet for the point set $P \cap \tau$. We present an algorithm constructing a net-restricted ST T for N_τ. Then we show that T is a net-restricted $(1+O(\epsilon \log \epsilon^{-1}))$-ST and its weight is $O(\mathsf{opt} \cdot \log^2(\epsilon^{-1}))$, where opt denotes the minimum weight of a net-restricted $(1+\epsilon)$-ST for $N_\tau \subset P$; and let OPT be one such a $(1+\epsilon)$-ST.

▶ **Lemma 20** (⋆). *Let $\mathsf{opt}(\tau)$ be the weight of the lightest net-restricted $(1+\epsilon)$-ST for $N_\tau \cup \{s\}$. There is a polynomial time algorithm returns a net-restricted $(1+\epsilon \cdot \log(\epsilon^{-1}))$ ST T for $N_\tau \cup \{s\}$ of weight $O(\log^2(\epsilon^{-1}) \cdot \mathsf{opt}(\tau))$.*

Shallow tree construction algorithm. Similar to Section 4, we assume that $\tau \subset [0,1] \times [-\sqrt{\epsilon}, \sqrt{\epsilon}]$ and $s = (2,0)$ (see Figure 4). For each point $p \in N_\tau$, we use the same set of vertical lines $L_0(p), L_1(p), \ldots$ defined in Equation (3). For each point p and each integer i, let $\mathcal{E}_p(i)$ denote the intersection of $\mathcal{E}_{ps}$ with the strip between the two lines $L_i(p)$ and $L_{i+1}(p)$ and $B_i(p)$ be the smallest axis-parallel rectangle containing $\mathcal{E}_p(i)$. The point p is associated with the rectangles $B_0(p), B_1(p), \ldots, B_k(p)$ with $k = O(\log \epsilon^{-1})$. A rectangle R *lies on* two distinct vertical lines L_1 and L_2 if two of its vertical sides are contained in L_1 and L_2.

We construct minimum hitting sets in k iterations. Starting from $i = 0$, let $\mathcal{R}_i$ be the set of rectangles $\{B_i(p) : p \in P_\tau\}$. Consider every pair of lines $L_i \in \mathcal{L}_i, L_{i+1} \in \mathcal{L}_{i+1}$ such that there are some rectangles in $\mathcal{R}_i$ lying on L_i and L_{i+1}. Let $\mathcal{R}_i(L_i, L_{i+1})$ be the set of rectangles in $\mathcal{R}_i$ lying on L_i and L_{i+1} that have nonempty intersection with P. We find the minimum hitting set of $\mathcal{R}_i(L_i, L_{i+1})$ from P. For each point u in the hitting set, let $\mathsf{iter}_p(u) = i$ for every $p \in P(L_i)$ (and define $\mathsf{iter}_p(s) := k+1$ for all $p \in P_\tau$).

We build a graph G on P as follows: For every point $p \in N_\tau$, let $s_0, s_1, \ldots s_h$ be a sequence of hitting points so that for $i = 0, 1, \ldots h$, if $B_i(p) \cap P \neq \emptyset$, then we choose an arbitrary point from the minimum hitting set of $\mathcal{R}(L_i(p), L_{i+1}(p))$ that hits $B_i(p)$ (otherwise no point is chosen from $B_i(p)$. We add the path $(p, s_0, s_1, \ldots, s_h, s)$ to G. The path $(p, s_0, s_1, \ldots, s_h, s)$ is called the *candidate path* of p in G.

For each point $p \in N_\tau$, let $H(p) = (p, s_0, s_1, \ldots s_h, s)$ be the candidate path of p in G. We prune the path $H(p)$ to eliminate some of the vertices: Initialize $H'(p) := H(p)$. While there exist two consecutive interior vertices v and v' in $H'(p)$ such that $\mathsf{iter}_p(v') = \mathsf{iter}_p(v) + 1$, then let the first such pair be v, v' and eliminate v' from $H'(p)$. Assume that $H'(p) = (p, s'_0, s'_1, \ldots s'_l, s)$ is the result of the pruning process. Observe that $H'(p)$ is a path from p to s. We call $H'(p)$ the *approximate path* of p. Let G' be the union of paths $H'(p)$ for all $p \in N_\tau$. We return the shortest-path tree $\mathsf{SPT}(G')$ of G' rooted at s.

▶ **Lemma 21** (⋆). *Let p be a point in N_τ, and let $\pi = (p, s_0, s_1, \ldots, s_{h-1}, s)$ be the approximate path of p in G', we have $\mathbf{w}(\pi) \leq (1 + \epsilon \log \epsilon^{-1})\, d(u, s)$.*

The proof of Lemma 21 is similar to the one with Steiner points. Since $\mathsf{SPT}(G')$ is the shortest-path tree of G', it contains ps-paths of weight $(1+O(\epsilon \log \epsilon^{-1}))\, d(p,s)$ for all $p \in N_\tau$.

Weight Analysis. We show that $\mathbf{w}(G) \leq O(\mathsf{opt} \cdot \log \epsilon^{-1})$, and then prove that $\mathbf{w}(G') \leq O(\mathbf{w}(G) \cdot \log \epsilon^{-1})$. Combined with the trivial bound $\mathbf{w}(\mathsf{SPT}(G')) \leq \mathbf{w}(G')$, we obtain $\mathbf{w}(\mathsf{SPT}(G')) \leq O(\mathsf{opt} \cdot \log^2 \epsilon^{-1}))$, as required.

Proof Overview. To bound the weight of G, we partition its edges into two types: local edges, which connect vertices in consecutive strips, and crossing edges, which connect vertices

in non-consecutive strips. The contribution of local edges can be bounded using the number of paths in OPT that intersect the upper strip. For crossing edges, we charge each such edge to an appropriate representative edge of OPT. The key observation is that if G contains a long edge e, then there must exist an edge $e' \in \mathsf{OPT}$ whose length is at least $\mathbf{w}(e)$, and we charge e to this e'. Because the lengths at different scale differ by exponential factors, each edge of OPT can be charged only by $O(\log(1/\varepsilon))$ edges of G at the same scale. Therefore, each such representative edge e' is charged at most $\log(1/\varepsilon)$ times, which yields the desired bound. The bound on the weight of G' follows from the fact that each consecutive edge can appear in at most $O(\log(1/\varepsilon))$ candidate paths in G.

References

1 Charles J Alpert, Te C Hu, Jen-Hsin Huang, Andrew B Kahng, and David Karger. Prim-dijkstra tradeoffs for improved performance-driven routing tree design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(7):890–896, 2002.

2 Baruch Awerbuch, Alan E. Baratz, and David Peleg. Cost-sensitive analysis of communication protocols. In *Proc. 9th ACM Symposium on Principles of Distributed Computing, (PODC)*, pages 177–187, 1990. doi:10.1145/93385.93417.

3 Baruch Awerbuch, Alan E. Baratz, and David Peleg. Efficient broadcast and light-weighted spanners. *manuscript*, 1991.

4 Yehuda Ben-Shimol, Amit Dvir, and Michael Segal. SPLAST: A novel approach for multicasting in mobile wireless ad hoc networks. In *Proc. 15th IEEE Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, volume 2, pages 1011–1015, 2004. doi:10.1109/PIMRC.2004.1373851.

5 Gengjie Chen and Evangeline F. Y. Young. SALT: provably good routing topology by a novel Steiner shallow-light tree algorithm. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 39(6):1217–1230, 2020. doi:10.1109/TCAD.2019.2894653.

6 Otfried Cheong and Changryeol Lee. Single-source dilation-bounded minimum spanning trees. *Int. J. Comput. Geom. Appl.*, 23(3):159–170, 2013. doi:10.1142/S0218195913500052.

7 Jason Cong, Andrew B. Kahng, Gabriel Robins, Majid Sarrafzadeh, and C. K. Wong. Performance-driven global routing for cell based ICs. In *Proc. IEEE International Conference on Computer Design: VLSI in Computer & Processors (ICCD)*, pages 170–173, 1991. doi:10.1109/ICCD.1991.139874.

8 Jason Cong, Andrew B. Kahng, Gabriel Robins, Majid Sarrafzadeh, and Chak-Kuen Wong. Provably good performance-driven global routing. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 11(6):739–752, 1992. doi:10.1109/43.137519.

9 Razvan Cristescu, Baltasar Beferull-Lozano, Martin Vetterli, and Roger Wattenhofer. Network correlated data gathering with explicit communication: Np-completeness and algorithms. *IEEE/ACM Trans. Netw.*, 14(1):41–54, 2006. doi:10.1145/1133553.1133557.

10 Ludwig Danzer and Branko Grünbaum. Intersection properties of boxes in $\mathbb{R}^d$. *Comb.*, 2(3):237–246, 1982. doi:10.1007/BF02579232.

11 Yefim Dinitz, Michael Elkin, and Shay Solomon. Low-light trees, and tight lower bounds for Euclidean spanners. *Discret. Comput. Geom.*, 43(4):736–783, 2010. doi:10.1007/S00454-009-9230-Y.

12 Michael Elkin and Shay Solomon. Narrow-shallow-low-light trees with and without Steiner points. *SIAM Journal on Discrete Mathematics*, 25(1):181–210, 2011. doi:10.1137/090776147.

13 Michael Elkin and Shay Solomon. Steiner shallow-light trees are exponentially lighter than spanning ones. *SIAM J. Comput.*, 44(4):996–1025, 2015. doi:10.1137/13094791X.

14 Stephan Held and Daniel Rotter. Shallow-light Steiner arborescences with vertex delays. In *Proc. 16th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 7801 of *LNCS*, pages 229–241. Springer, 2013. doi:10.1007/978-3-642-36694-9_20.

15 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985. doi:10.1145/2455.214106.

16 Xiaohua Jia, Ding-Zhu Du, Xiao-Dong Hu, Man-Kei Lee, and Jun Gu. Optimization of wavelength assignment for QoS multicast in WDM networks. *IEEE Trans. Commun.*, 49(2):341–350, 2001. doi:10.1109/26.905896.

17 Ardalan Khazraei and Stephan Held. An improved approximation algorithm for the uniform cost-distance Steiner tree problem. In *Proc. 18th Workshop on Approximation and Online Algorithms (WAOA)*, volume 12806 of *LNCS*, pages 189–203, 2020. doi:10.1007/978-3-030-80879-2_13.

18 Samir Khuller, Balaji Raghavachari, and Neal E. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–321, 1995. Preliminary version at SODA 1993. doi:10.1007/BF01294129.

19 Vachaspathi P. Kompella, Joseph Pasquale, and George C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Trans. Netw.*, 1(3):286–292, 1993. doi:10.1109/90.234851.

20 Dejan Kostic and Amin Vahdat. Latency versus cost optimizations in hierarchical overlay networks. Technical report, Technical report, Duke University,(CS-2001-04), 2002.

21 Wei Li, Yuxiao Qu, Gengjie Chen, Yuzhe Ma, and Bei Yu. Treenet: Deep point cloud embedding for routing tree construction. In *Proc. 26th Asia and South Pacific Design Automation Conference (ASPDAC)*, pages 164–169. ACM Press, 2021. doi:10.1145/3394885.3431566.

22 Weifa Liang and Yuzhen Liu. Online data gathering for maximizing network lifetime in sensor networks. *IEEE Trans. Mob. Comput.*, 6(1):2–11, 2007. doi:10.1109/TMC.2007.250667.

23 Hong Luo, Jun Luo, Yonghe Liu, and Sajal K Das. Adaptive data fusion for energy efficient routing in wireless sensor networks. *IEEE Transactions on Computers*, 55(10):1286–1299, 2006. doi:10.1109/TC.2006.157.

24 Frank Nielsen. Fast stabbing of boxes in high dimensions. *Theor. Comput. Sci.*, 246(1-2):53–72, 2000. doi:10.1016/S0304-3975(98)00336-3.

25 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000. doi:10.1137/1.9780898719772.

26 F. Sibel Salman, Joseph Cheriyan, R. Ravi, and S. Subramanian. Buy-at-bulk network design: Approximating the single-sink edge installation problem. In *Proc. 8th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 619–628, 1997. URL: http://dl.acm.org/citation.cfm?id=314161.314397.

27 Anees Shaikh and Kang Shin. Destination-driven routing for low-cost multicast. *IEEE Journal on Selected Areas in Communications*, 15(3):373–381, 1997. doi:10.1109/49.564135.

28 Hanan Shpungin and Michael Segal. Near-optimal multicriteria spanner constructions in wireless ad hoc networks. *IEEE/ACM Transactions on etworking*, 18(6):1963–1976, 2010. doi:10.1109/TNET.2010.2053381.

29 Shay Solomon. Euclidean Steiner shallow-light trees. *J. Comput. Geom.*, 6(2):113–139, 2015. doi:10.20382/JOCG.V6I2A7.

30 Jürgen Vogel, Jörg Widmer, Dirk Farin, Martin Mauve, and Wolfgang Effelsberg. Priority-based distribution trees for application-level multicast. In *Proceedings of the 2nd Workshop on Network and System Support for Games (NETGAMES)*, pages 148–157. ACM, 2003. doi:10.1145/963900.963914.

31 Pascal Von Rickenbach and Rogert Wattenhofer. Gathering correlated data in sensor networks. In *Proceedings of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, pages 60–66, 2004. doi:10.1145/1022630.1022640.

32 Bang Ye Wu, Kun-Mao Chao, and Chuan Yi Tang. Light graphs with small routing cost. *Networks*, 39(3):130–138, 2002. doi:10.1002/NET.10019.

Topological Simplification Guided by Forbidden Regions

Jakub Leśkiewicz[1]
Jagiellonian University, Kraków, Poland

Bartosz Furmanek
Jagiellonian University, Kraków, Poland

Michał Lipiński
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Dmitriy Morozov
Lawrence Berkeley National Laboratory, CA, USA

Abstract

Topological simplification is the process of reducing complexity of a function while maintaining its essential features. Its goal is to find a new filter function, which reorders cells of the input complex in a way which eliminates some persistent homological features, without affecting the rest. We present a new approach to simplification based on the concept of *forbidden regions* and combinatorial dynamics. It allows us to reorder and cancel critical values, whose cancellation is not possible using existing methods because they are not consecutive in the total order. Each such cancellation takes $O(c \cdot n)$ time in the worst case, where c is the number of birth-death pairs and n is the size of the input complex.

2012 ACM Subject Classification Computing methodologies → Algebraic algorithms; Mathematics of computing → Algebraic topology

Keywords and phrases persistent homology, topological simplification, depth posets

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.72

Related Version *Full Version*: `https://arxiv.org/abs/2603.16416` [23]

Funding *Jakub Leśkiewicz*: The research was partially funded by the Polish National Science Center under Opus Grant No. 2019/35/B/ST1/00874 and Opus Grant 2025/57/B/ST1/00550.
Bartosz Furmanek: The research was partially funded by the Polish National Science Center under Opus Grant No. 2019/35/B/ST1/00874 and Opus Grant 2025/57/B/ST1/00550.
Michał Lipiński: This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 101034413.
Dmitriy Morozov: This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Contract No. DE-AC02-05CH11231.

Acknowledgements Jakub Leśkiewicz wants to thank his supervisor, Prof. Marian Mrozek, for scientific guidance, patience, and opportunity to delay the rest of his duties while writing this work. The author also extends thanks to his entire family, to Zuzanna Świątek, and to Mikołaj Kardyś, BEng, MSc, for providing meals during the most intensive periods of work.

1 Introduction

Simplification of real-valued functions is one of the central topics in Morse theory. In the classical (smooth) setting, one of the most notable examples of such simplifications is performed throughout the proof of the h-cobordism theorem [25, 24]. In the discrete

[1] Corresponding author

© Jakub Leśkiewicz, Bartosz Furmanek, Michał Lipiński, and Dmitriy Morozov;
licensed under Creative Commons License CC-BY 4.0

setting, Forman's theory [18, 17] studies the reversal of a *unique* combinatorial path between two critical points as a way of reducing the total number of critical points and, therefore, simplifying the Morse complex.

More recently, simplification has been studied in the context of persistent homology. The authors of [14] introduced the problem of persistence-sensitive simplification – asking to simplify all pairs with persistence below a given threshold – and gave an algorithm for 2-manifolds. Their solution was later improved to linear time [1]. Another approach, based on Forman's combinatorial vector fields, was presented in [3]. This work drew connection to the cancellation procedure by observing that whenever the unique path connects two critical points with locally lowest difference in function values, their cancellation does not affect the remaining part of the persistence diagram [3]. These *apparent pairs* have also been called *close pairs* [9] and *shallow pairs* [15].

This observation further helped in optimization of (persistent) homology computation [2] and shape reconstruction [4]. The idea of pruning pairs of critical cells, following Forman's approach, has been extensively studied in data visualization [6, 7, 10, 16, 21, 22, 27]. Recent works on topological optimization [20, 26] offer an alternative, albeit less controlled approach to simplification.

But there remains a critical gap. The works that are able to rigorously control the changes in persistent homology [14, 1, 3] are only able to simplify 0-dimensional persistent homology (as well as codimension-1, by duality). Meanwhile, the middle dimensions – e.g., 1-dimensional homology on 3-manifolds – are important in practice.

In this work, we study how relations between persistence pairs calculated by the standard lazy reduction algorithm [11, 5] can guide us in simplifying a discrete Morse function h, while controlling the changes in its persistence diagram *in any dimension* and the overall gradient structure. These relations organize persistence pairs in a hierarchical structure called a *depth poset* [12]. As observed in [13, 26], these relations describe the obstacles to modifying a function without changing its persistence diagram.

Concretely, for a given persistence pair α, we define *forbidden regions* for its death and birth cells, which describe the parts of the persistence diagram that α cannot move to without changing the persistence pairing. Conversely, when the forbidden regions leave a gap – a path from α to the diagonal – we can construct a homotopy that brings α to the diagonal without changing the rest of the persistence pairs and the gradient structure. This allows us to identify a broader family of persistence pairs, possibly with high persistence, that can be safely and selectively removed. We summarize our main contribution in the following theorem, where $\mathrm{BD}(h)$ denotes the set of birth-death pairs induced by a discrete Morse function h, and $\mathrm{Crit}(\mathcal{V}_h)$, the set of critical cells for h.

▶ **Theorem 1.** *Let h be a discrete Morse function on a Lefschetz complex X. If $\alpha \in \mathrm{BD}(h)$ is a persistence pair such that forbidden regions of its death and birth cells do not intersect, and there exists exactly one gradient path between the paired critical cells, then there exists a discrete Morse function h' on X such that $\mathrm{BD}(h') = \mathrm{BD}(h) \setminus \{\alpha\}$ and $h(x) = h'(x)$ for all $x \in \mathrm{Crit}(\mathcal{V}_{h'})$.*

We present a constructive proof to this theorem, which provides an algorithm explicitly tracking all changes in relations throughout the homotopy and the final path reversal. As a result, we obtain already computed relations between pairs in $\mathrm{BD}(h')$, which enables iterative simplification.

2 Preliminaries

▶ **Definition 2.** *A* Lefschetz complex *is a triplet* $(X, \dim, D)$*, where* X *is a finite set of elements called cells,* $\dim : X \to \mathbb{N}$ *is a map assigning a dimension to each cell, and* $D : X \times X \to \mathbb{Z}_2$ *is the boundary coefficient such that* $D(x, y) \neq 0$ *implies* $\dim x + 1 = \dim y$*, in which case we say* x *is a* facet *of* y*. Additionally, we require that for any* $x, y \in X$ *we have* $\sum_{z \in X} D(x, z) \cdot D(z, y) = 0$*. We also define the coboundary coefficient as* $D^{\perp}(y, x) \coloneqq D(x, y)$*.*

Lefschetz complexes generalize simplicial, cubical, and cellular complexes while remaining concrete enough to define persistent homology. When it does not lead to confusion, we shorten the notation and refer to the set of cells, X, as the Lefschetz complex. We often interpret D and $D^{\perp}$ as a matrix, in which case we put the arguments in the square brackets for emphasis, e.g., $D[x, y]$. We write X_n for the set of n-dimensional cells of X, and D_n, $D_n^{\perp}$ for the n-th boundary and coboundary matrix, respectively.

▶ **Definition 3** (Discrete Morse function)**.** *Let* X *be a Lefschetz complex. A map* $h : X \to \mathbb{R}$ *is called a* discrete Morse function *(dMf, for short) if the following conditions are satisfied for all* $x, y \in X$*.*

- **(i)** *if* $D(x, y) = 1$ *then* $h(x) \leq h(y)$ *(weak monotonicity),*
- **(ii)** *if* $h(x) = h(y)$ *then either* $D(x, y) = 1$ *or* $D(y, x) = 1$ *(pairing),*
- **(iii)** *for every* $y \in \mathbb{R}$*, we have* $\#h^{-1}(y) \leq 2$ *(almost injective).*

In particular, we say that X is *filtered* by h. It also induces an h-order on X:

$$x <_h y \iff \big(h(x) < h(y)\big) \text{ or } \big(h(x) = h(y) \text{ and } \dim x < \dim y\big).$$

If X is filtered by a dMf h, then we always assume that rows and columns of D_n are ordered by the h-order, and those of $D_n^{\perp}$ by the reversed h-order.

To calculate persistent homology, we use the original version of the persistence algorithm [11], called *lazy reduction algorithm*, described in the form we need in [26]. The algorithm relies on an auxiliary function low, which, for a given column, returns the index of the row containing the lowest non-zero entry in that column. For a given (co)boundary matrix D_n, Algorithm 1 performs successive column additions, which results in a decomposition $D_n = R_n U_n$ with U_n invertible and upper triangular. Moreover, if $x \neq y$ and $U_n[x, y] \neq 0$, then column $R_n[:, x]$ was added to $R_n[:, y]$ by the algorithm. Observe that in D_n the rows are indexed by $(n-1)$-dimensional cells and the columns by n-dimensional cells. The same holds for R_n; however, both the rows and columns of U_n are indexed by n-dimensional cells. Similarly, we obtain $D^{\perp} = R^{\perp} U^{\perp}$ decomposition by applying $D^{\perp}$ to the algorithm.

Algorithm 1 Lazy reduction of the matrix over $\mathbb{Z}_2$.

```
R_n=D_n,   U_n=I
for y over the columns of R_n (left to right) do
    while R_n[:, y] ≠ 0 and there exists a preceding column x with
      low(R_n[:, x]) = low(R_n[:, y]) do
        R_n[:, y] ← R_n[:, y] + R_n[:, x]
        U_n[x, :] ← U_n[x, :] + U_n[y, :]
```

We say that $\alpha = (\alpha^{\circ}, \alpha^{\times})$ is an (n-dimensional) *birth-death* pair if α° is a low of $R_{n+1}[:, \alpha^{\times}]$. α is an (n-dimensional) birth-death pair if and only if $\alpha^{\times}$ is a low of $R_n^{\perp}[:, \alpha^{\circ}]$ [8]. We refer to α° and $\alpha^{\times}$ as *birth* and *death* cells, respectively. The *dimension* of a birth-death pair is

the dimension of its birth cell. We denote the set of all birth-death pairs by $\mathrm{BD}(h)$ and the set of all n-dimensional birth-death pairs by $\mathrm{BD}_n(h)$. The cells in dimension n that are not paired at all – their columns in R_n are zero, and there are no columns in R_{n+1} that have them as the lowest non-zero entry – are *n-dimensional homology generators*. It is convenient to assume that these generators also belong to some birth-death pair, even if its second component is undefined.

Let $x, y \in X_n$. If $U_n[x, y] = 1$, we say that there is a *homological relation* between x and y and denote this fact by $x \xrightarrow{\times} y$. Dually, if $U_n^{\perp}[x, y] = 1$, we write $x \xrightarrow{\circ} y$ to indicate a *cohomological relation*. If the relation type is not important, we simply write $x \to y$. If x and y are unrelated, we write $x \nrightarrow y$, adding a superscript to specify the missing relation type, e.g., $x \overset{\times}{\not\to} y$ if $U[x, y] = 0$. Observe that if $x \xrightarrow{\times} y$, then x must be a death cell, whereas y may be either a death cell or birth cell. Similarly if $x \xrightarrow{\circ} y$ then x has to be a birth cell, while the type of y remains unspecified. We extend these notions to birth-death pairs: $\beta \xrightarrow{\times} \alpha$ whenever $\beta^{\times}$ is homologically related to any component of α; similarly for other kinds of arrows. As rows and columns of D_n and $D_n^{\perp}$ are ordered with respect to the h-order and the reversed h-order, so are R_n, $R_n^{\perp}$, U_n and $U_n^{\perp}$. We emphasize that $R_n^{\perp}$ and $U_n^{\perp}$ are not transposed matrices R_n and U_n, but components of lazy decomposition $D_n^{\perp} = R_n^{\perp} U_n^{\perp}$.

The persistence diagram is a set of two dimensional points $(h(\alpha^{\circ}), h(\alpha^{\times}))$ for $\alpha \in \mathrm{BD}(h)$. When we visualize a persistence diagram (see Figure 2), it is convenient to add the *diagonal*, i.e., all points (x, x) for $x \in \mathbb{R}$, and to annotate the arrows with the type of the relation. Since a dMf is not injective in general, it can generate birth-death pairs on the diagonal of the persistence diagram. We denote the set of such diagonal pairs by $\overline{\mathrm{BD}}(h)$ and use notation $\hat{\mathrm{BD}}(h)$ for the pairs above the diagonal.

▶ **Observation 4.** *If pairs $\alpha, \beta \in \mathrm{BD}_n(h)$ and $\beta \to \alpha$, then $h(\beta^{\circ}) > h(\alpha^{\circ})$ and $h(\beta^{\times}) < h(\alpha^{\times})$. (see [26, Lemma 2.1])*

The preceding observation means that if we depict every relation between the birth-death pairs of the same dimension as arrows in the persistence diagram, then every such arrow points up and to the left.

▶ **Definition 5.** *Let X be a Lefschetz complex and let $h\colon X \to \mathbb{R}$ be a dMf. A* topological simplification of h *is a discrete Morse function h' such that $\hat{\mathrm{BD}}(h') \subset \hat{\mathrm{BD}}(h)$ and $h'(\alpha^{\circ}) = h(\alpha^{\circ})$ and $h'(\alpha^{\times}) = h(\alpha^{\times})$, whenever $\alpha \in \hat{\mathrm{BD}}(h')$.*

In words, a topological simplification removes some off-diagonal persistence pairs and preserves the rest.

▶ **Definition 6.** *A* combinatorial vector field *(or a vector field, for short) on a Lefschetz complex X is a partition $\mathcal{V}$ of X into singletons, called* critical cells*, and facet–cofacet pairs, called* vectors*. $\mathrm{Crit}(\mathcal{V})$ denotes the family of all critical cells of $\mathcal{V}$; $\mathrm{Vec}(\mathcal{V})$, the family of all vectors. We use the convention that the dimension of a vector is the smaller dimension of its two components.*

A combinatorial vector field $\mathcal{V}$ induces a digraph $G_{\mathcal{V}} = (X, E)$. Every edge $(x, y) \in E$ is either an *explicit arc* when $(x, y) \in \mathcal{V}$ or an *implicit arc* when $D(y, x) = 1$ and $(x, y) \notin \mathcal{V}$. A path ρ has *dimension* k if it consists only of cells of dimension k and $k + 1$. In particular, any path from y to x, where $x, y \in \mathrm{Crit}(\mathcal{V})$ and $k = \dim(x) = \dim(y) - 1$ is of dimension k and alternates between k and $k + 1$ dimensional cells.

A combinatorial vector field $\mathcal{V}$ is called *gradient* if $G_{\mathcal{V}}$ is acyclic. If there is a path between vertices y and x, then we write $y \overset{\mathcal{V}}{\rightsquigarrow} x$, omitting the superscript when the vector field is clear from the context. $\mathcal{V}^k$ denotes the union of all k-dimensional vectors and critical

cells of dimension k and $k+1$. Finally, note that for a given discrete Morse function h, the non-empty preimages $\mathcal{V}_h \coloneqq \{h^{-1}(a) \mid a \in \mathbb{R},\ h^{-1}(a) \neq \emptyset\}$ form a combinatorial gradient vector field.

The *Morse complex* connects homological and dynamical perspectives on scalar functions. It is not required to carry out the reasoning we need, but it will simplify it considerably.

▶ **Definition 7** (Morse complex). *Let $\mathcal{V}$ be a combinatorial gradient vector field on X. The* Morse complex *of $\mathcal{V}$ is a Lefschetz complex, denoted by $\mathcal{M}(\mathcal{V})$, consisting of the set of critical cells of $\mathcal{V}$ along with the restriction of* dim*. The boundary coefficient $D_{\mathcal{M}}(x,y)$ is given by the number of paths in $G_{\mathcal{V}}$ from y to $x \pmod 2$, provided $\dim y = \dim x + 1$ and 0 otherwise.*

The most useful properties of Morse complexes for our work is that they describe the off-diagonal birth-death pairs. Indeed, if $\mathcal{V}_h$ is a gradient vector field of some dMf h, its restriction to $\mathcal{M}(\mathcal{V}_h)$, denoted by $h_{\mathcal{M}}$, is an injective dMf. The next observation follows from [15, Theorem 4.3].

▶ **Corollary 8.** *Let X be filtered by dMf h. Then $\mathrm{BD}(h_{\mathcal{M}}) = \hat{\mathrm{BD}}(h)$ and $U_{\mathcal{M}}$, $U_{\mathcal{M}}^{\perp}$ are restrictions of U and $U^{\perp}$ to the critical cells.*

It follows that we can identify the components of the pairs in $\hat{\mathrm{BD}}(h)$ with elements of $\mathrm{Crit}(\mathcal{V}_h)$, while the vectors are the diagonal pairs, $\mathrm{Vec}(\mathcal{V}_h) = \overline{\mathrm{BD}}(h)$. This perspective enables us to apply the following classical theorem.

▶ **Theorem 9** ([19, Theorem 9.1]). *Let x be a k-dimensional critical cell and y be a $k+1$ dimensional critical cell of a gradient vector field $\mathcal{V}$. If there exists a unique path ρ from y to x, then reversing it in $\mathcal{V}$ produces another gradient vector field, which we denote $\mathcal{V}^{-\rho}$. The critical cells of $\mathcal{V}^{-\rho}$ are exactly the critical cells of $\mathcal{V}$ apart from x and y.*

We say that $\alpha \in \hat{\mathrm{BD}}(h)$ is *reversible* if there exists exactly one path between $\alpha^{\times}$ and α° in $\mathcal{V}_h$. However, the theorem alone gives no guarantee that elimination of a pair of critical cells will not affect the remaining pairs in $\hat{\mathrm{BD}}(h)$. Identifying those pairs that can be safely removed is therefore a key challenge.

▶ **Definition 10.** *Let X be a Lefschetz complex filtered by dMf h. A pair $(x,y) \in X \times X$ such that $D(x,y) = 1$ is a* shallow pair *if $h(x)$ is the maximum among facets of y and $h(y)$ is the minimum among cofacets of x.*

Observe that every shallow pair is a birth-death pair. Shallow pairs were introduced as apparent pairs in [2] and as close pairs in [9]. Since the theory behind them was later developed in the framework of the *depth posets* [13], we adopt the name from that setting. Shallow pairs are closely related to an algebraic operation called *Lefschetz cancelation*.

▶ **Definition 11.** *Let $(s,t) \in X \times X$ be a pair in a Lefschetz complex such that s is a facet of t. A* cancellation *of (s,t) produces a* quotient*, another Lefschetz complex $(\hat{X}, \hat{\dim}, \hat{D})$ such that $\hat{X} = X \setminus \{s,t\}$, $\hat{\dim}$ is a restriction of* dim *to $\hat{X}$ and $\hat{D}(x,y) = D(x,y) + D(s,y) \cdot D(x,t)$.*

The boundary map in the quotient can be written in matrix form: if $\dim t = n$ and $\hat{D}_n$ is the n-th boundary matrix of $\hat{X}$, then $\hat{D}_n[:,y] = D_n[:,y] + D_n[s,y] \cdot D_n[:,t]$, after erasing row s and column t. Throughout this paper, we often refer to small modifications of matrices based on their previous state. In such cases, any matrix M after modification is denoted $\hat{M}$.

▶ **Theorem 12** ([12, Theorem 3.2]). *Let X be filtered by a dMf h. Fix a shallow pair α. Then birth-death pairs of quotient of X after Lefschetz cancelation of α are exactly $\mathrm{BD}(h) \setminus \{\alpha\}$, and every shallow pair of h distinct from α remains shallow in the quotient, which may in addition contain new shallow pairs not present in X.*

In other words, performing a Lefschetz cancellation on a shallow pair does not change the pairing between the rest of the cells. It is convenient to characterize shallow pairs in terms of the relations between cells.

▶ **Observation 13.** *An n-dimensional birth-death pair α is shallow if and only if $U_{n+1}[:, \alpha^\times]$ and $U_n^\perp[:, \alpha^\circ]$ are zero except $U_{n+1}[\alpha^\times, \alpha^\times]$ and $U_n^\perp[\alpha^\circ, \alpha^\circ]$. Equivalently, α is shallow iff $\beta \nrightarrow \alpha$ for any birth-death pair β.*

It is important to note that a Lefschetz cancellation leaves intact not only the pairing, but also the relations between cells.

▶ **Theorem 14.** *Let D_n be a boundary matrix, and $\hat{D}_n$, a boundary matrix of the quotient after cancellation of the $(n-1)$-th dimensional shallow pair α. Let R_nU_n and $\hat{R}_n\hat{U}_n$ be their respective decompositions obtained via the lazy reduction. Then, $U_n[x, y] = \hat{U}_n[x, y]$ for all x, y different than $\alpha^\times$. Moreover, symmetrically $U_{n-1}^\perp[x, y] = \hat{U}_{n-1}^\perp[x, y]$ for all x, y different than α°.*

The above theorem can be rephrased as follows.

▶ **Observation 15.** *Let X be filtered by a dMf and α be a shallow birth-death pair. If $\beta \to \gamma$ in X, then the same relation holds in the quotient $\hat{X}$ obtained after the cancellation of α, for all $\beta \neq \alpha$.*

Due to Observation 13 above, we introduce *critical shallow pairs*. An off-diagonal birth-death pair α is a critical shallow pair if there does not exist an off-diagonal birth-death pair β such that $\beta \to \alpha$. It is easy to see that critical shallow pairs are exactly the shallow pairs of the Morse complex, although they need not be shallow pairs in the original complex. Equivalently, a pair α is critically shallow if for every $\beta \to \alpha$, the pair β is a vector in $\mathcal{V}_h$.

▶ **Theorem 16.** *Let $\mathcal{V}$ be a combinatorial vector field on X and let $s, t \in \mathrm{Crit}(\mathcal{V})$ be such that $\dim s + 1 = \dim t$. Assume that there exists a unique path ρ from t to s. Then $\mathcal{M}(\mathcal{V}^{-\rho})$ is isomorphic to the quotient of $\mathcal{M}(\mathcal{V})$ after cancelling the pair (s, t). (See example in Figure 1.)*

So to find a topological simplification of h, one can find a critical shallow pair α that is reversible. Then, one has to invert the unique path ρ between $\alpha^\times$ and α°, and find h' with the property that $\mathcal{V}^{-\rho} = \mathcal{V}_{h'}$ and $h_{|\,\mathrm{Crit}(\mathcal{V}^{-\rho})} = h'_{|\,\mathrm{Crit}(\mathcal{V}^{-\rho})}$. Unfortunately, it may happen that there is no pair that is both shallow and reversible. One of the goals of this paper is to remedy this problem.

3 Homology and cohomology relations in the filter

To understand how birth-death pairs and the relationships between them change during changes of the dMf, one must study how they change upon transposition of two adjacent cells in the boundary matrix. This problem is well-studied; see [5] and [13]. Observing that the depth poset can be constructed from the union of homological and cohomological relations between birth-death pairs (see Theorem 4.8 in [12]), we reformulate the results from [13] in the language of this paper. First, we introduce two additional objects.

▶ **Lemma 17** ([13, Lemma 3.2])**.** *Fix a birth-death pair $\alpha \in \mathrm{BD}(h)$. If we remove all birth-death pairs below and to the right of α – in the region $(h(\alpha^\circ), +\infty] \times [-\infty, h(\alpha^\times))$ – by iteratively canceling shallow pairs, we get the same boundary matrix regardless of the order of cancellations.*

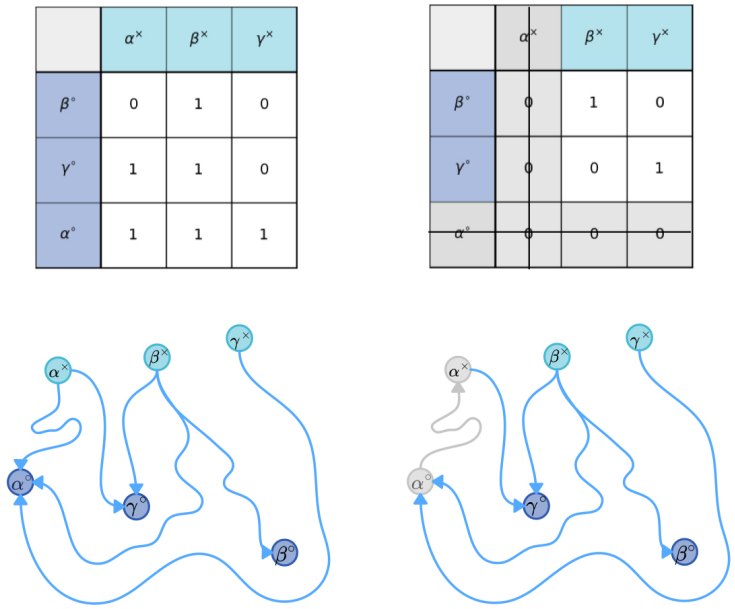


Figure 1 Two vector fields differing by a reversal of the path between components of a birth-death pair α. Critical cells are shown with colored nodes, and arrows between them symbolize paths created by vectors. Above each vector field is the boundary matrix of the corresponding Morse complex. Reversing the path between components of α gives the same boundary matrix as performing the Lefschetz cancelation.

Lemma 17 proves that the following definition is unambiguous.

▶ **Definition 18.** *Fix $\alpha, \beta \in \mathrm{BD}_n(h)$ such that the components of these pairs are consecutive columns in D_{n+1} or in $D_n^\perp$. Define $D_{n+1}^{\alpha,\beta}$ to be the matrix obtained by performing Lefschetz cancellations, always canceling shallow pairs, for all pairs lying in the bottom-right quadrant of α (excluding β if it eventually lies in this region) and for all pairs lying in the bottom-right quadrant of β (excluding α if it eventually lies in this region).*

Note that in the above definition, we can cancel all pairs in the bottom-right quadrants because, from Observation 4, there is no $\gamma \in \mathrm{BD}_n(h)$ such that $\beta \to \gamma \to \alpha$. After the cancellations, we have either (i) both α and β are shallow, or (ii) $\beta \to \alpha$ and β is shallow, or (iii) $\alpha \to \beta$ and α is shallow.

Now we are ready to utilize results from [13] in a series of theorems.

▶ **Theorem 19** (Result of death-cells transposition [13, Lemma 3.4])**.** *Let α, β be n-dimensional birth-death pairs such that $h(\alpha^\circ) < h(\beta^\circ)$. Then the transposition of $\alpha^\times$ and $\beta^\times$ does not change the values in U_{n+1}, while the changes in $U_n^\perp$ follow these rules:*

(1) *If $\beta \overset{\times}{\not\to} \alpha$ and $\left(\beta \overset{\circ}{\to} \alpha \text{ or } D_{n+1}^{\alpha,\beta}[\alpha^\circ, \beta^\times] = 1\right)$, then the pairing is unaffected and the row β° of the matrix $U_n^\perp$ changes according to the formula:*

$$\hat{U}_n^\perp[\beta^\circ, :] = U_n^\perp[\beta^\circ, :] + U_n^\perp[\alpha^\circ, :], \tag{1}$$

(2) *If $\beta \overset{\times}{\to} \alpha$, then the pairs $(\alpha^\circ, \alpha^\times), (\beta^\circ, \beta^\times)$ turn into $(\alpha^\circ, \beta^\times)$ and $(\beta^\circ, \alpha^\times)$ and $U_n^\perp$ changes as in (1).*

(3) *Otherwise, $U_n^\perp$ and the pairing remain unchanged.*

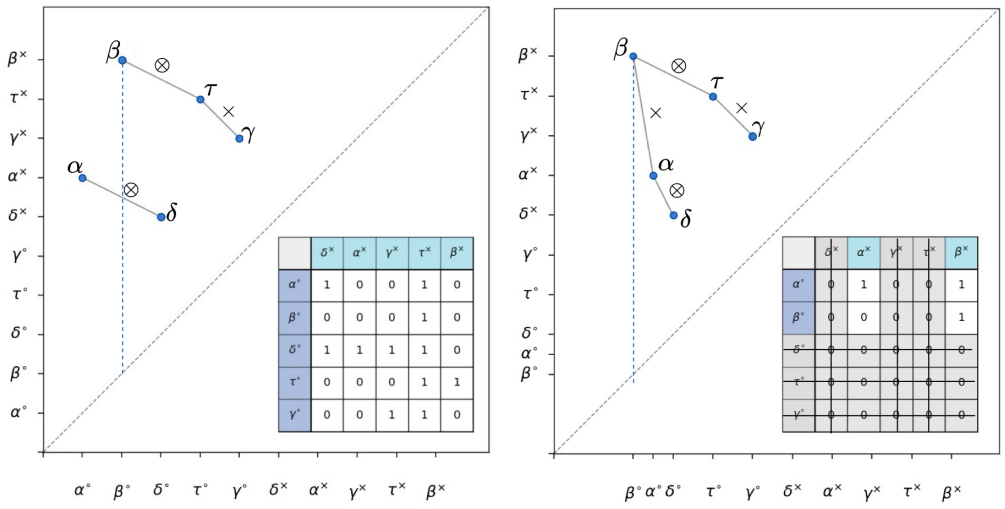


Figure 2 Left: $(n-1)$-st dimensional persistence diagram of some complex X, with D_n in the bottom-right corner. In the diagram, we denote by $\times$ homological relations between pairs, and by $\otimes$ relations which are homological and cohomological at the same time. To decide if moving β° past α° changes the relations between cells, as determined by Equation (1) in Theorem 20, we need to calculate $D_n^{\alpha,\beta}$. Right: The persistence diagram with the updated relation after the transposition of α° and β°. In the bottom-right corner, we show $D_n^{\alpha,\beta}$ before the transposition. The cells deleted by the Lefschetz cancellations are crossed out.

▶ **Theorem 20** (Result of birth-cells transposition [13, Lemma 3.3])**.** *Let α and β be n-dimensional birth-death pairs such that $h(\beta^\times) < h(\alpha^\times)$. Then the transposition of α° and β° does not change the values in $U_n^\perp$, while the changes in U_{n+1} follow these rules:*

(1) *If $\beta \not\xrightarrow{\circ} \alpha$ and $\left(\beta \xrightarrow{\times} \alpha \text{ or } D_{n+1}^{\alpha,\beta}[\beta^\circ, \alpha^\times] = 1\right)$, then the pairing is unaffected and the row $\beta^\times$ of the matrix U_{n+1} changes according to the formula:*

$$\hat{U}_{n+1}[\beta^\times, :] = U_{n+1}[\beta^\times, :] + U_{n+1}[\alpha^\times, :] \tag{2}$$

(2) *If $\beta \xrightarrow{\circ} \alpha$, then U_{n+1} changes as in (2), and pairs $(\alpha^\circ, \alpha^\times), (\beta^\circ, \beta^\times)$ turn into $(\beta^\circ, \alpha^\times)$ and $(\alpha^\circ, \beta^\times)$.*

(3) *Otherwise U_{n+1} and the pairing remain unchanged.*

▶ **Theorem 21** (Result of birth-death transposition [13, Lemma 3.5])**.** *A transposition between a birth and a death cell, which is a result of increasing birth, or decreasing death does not affect pairing or relations between birth-death pairs.*

Figure 2 presents an example of how a transposition affects the relationship between birth-death pairs. Now we introduce our own propositions, which will be useful later.

▶ **Proposition 22.** *Fix a pair $\alpha \in \mathrm{BD}_{(n-1)}(h)$. A transposition that increases the value of α° or decreases the value of $\alpha^\times$ and does not cause a switch cannot create a relation $\beta \to \alpha$ for any pair β.*

Note that a transposition may involve skipping two columns and rows when bypassing a combinatorial vector. The following proposition helps decrease complexity of the final algorithm.

▶ **Proposition 23.** *Let h, h' be two dMfs such that $\mathcal{V}_h = \mathcal{V}_{h'}$ and the difference between h-order and h'-order is a transposition between a critical cell and a vector. Then h and h' generate the same off-diagonal birth-death pairs and relations between them.*

Proof. As this process does not change the (co)boundary matrix of $\mathcal{M}(\mathcal{V}_h)$, it cannot change the pairing or relations between critical cells. ◀

Finally, the following two corollaries give us an opportunity to focus only on specific cases during the construction of the homotopy below.

▶ **Corollary 24.** *Take a pair $\alpha \in \mathrm{BD}_n(h)$. If x is a cell such that $\alpha^\circ <_h x$ and also $x \xrightarrow{\circ} \alpha^\circ$, then x is an n-dimensional birth cell. Analogously, if $y <_h \alpha^\times$ and $y \xrightarrow{\times} \alpha^\times$, then y is an $(n+1)$-dimensional death cell.*

Proof. Because $y \xrightarrow{\times} \alpha^\times$, the column indexed by y was added to column $\alpha^\times$ during the lazy reduction of matrix D_{n+1}. Because lazy reduction never adds zero columns, column y in R_{n+1} has a unique low, so it is a death cell. Analogously, if $x \xrightarrow{\circ} \alpha^\circ$, then column x was added to column α° in $D_n^\perp$, so x is a birth cell. ◀

4 Constructing the homotopy

4.1 Homotopy

Recall that a linear homotopy between two maps f_0 and f_1 is a family of maps $f_t(x) := H(t,x) = (1-t)f_0(x) + tf_1(x)$ for $t \in [0,1]$.

We say that $A \subset X$ is *connected* if it Hasse diagram – the graph whose vertices are the cells of A with an edge for every boundary relation – is connected. An *f-induced partition* is a partition $\mathcal{A}$ of X into maximal, with respect to inclusion, sets A, such that f is constant on A, and every A is connected.

▶ **Theorem 25.** *Let f_0 and f_1 be two dMfs defined on X such that $\mathcal{V}_{f_0} = \mathcal{V}_{f_1}$. Let $f_t(x) := H(t,x)$ be the linear homotopy between f_0 and f_1. Let $\mathcal{V}_{f_t}$ denote the f_t-induced partition of X. Then, $\mathcal{V}_{f_t} = \mathcal{V}_{f_0}$ for every $t \in [0,1]$.*

Using this theorem, we can represent our homotopy as a finite series of transpositions, allowing us to analyze only a finite number of time steps. Indeed, along a homotopy $(f_t)_{t\in[0,1]}$ there are only finitely many parameters t at which f_t fails to be a dMf. On each open interval between two such parameters, the induced f_t-order is well-defined and remains constant (in particular, it does not depend on t). Consequently, for a sufficiently fine discretization of $[0,1]$, consecutive f_t-orders differ by exactly one transposition.

4.2 Journey to the diagonal

Consider an example in Figure 3 and assume that our goal is to reduce the lifetime of the pair α to be arbitrarily small, without changing the pairing or the vector field. To reduce the lifetime, we may increase the value of α° and decrease the value of $\alpha^\times$, along with a set of vectors. We may implement this as a series of "moves" of the birth-death pair to the right and down in the persistence diagram.

Unfortunately, our moves are constrained: if we want to preserve the original vector field, then we cannot decrease $\alpha^\times$ below β° as $\alpha^\times \rightsquigarrow \beta^\circ$, and similarly, α° cannot increase above $\gamma^\times$. Moreover, as $\beta^\times \xrightarrow{\times} \alpha^\times$ and $\gamma^\times \xrightarrow{\times} \alpha^\times$, we also cannot decrease $\alpha^\times$ below these levels, without switches in pairing. Even worse, because $\xi^\circ \xrightarrow{\circ} \alpha^\circ$, α° cannot increase above ξ° without another switch.

This appears to be a serious obstacle. However, when we examine the persistence diagram (see bottom part of the Figure 3), we notice, following Observation 4, that increasing α° above β° breaks both homological relations of $\alpha^\times$ without changing the pairing. Afterwards, we are able to decrease $\alpha^\times$ as close to α° as we want. This motivates our central notion of forbidden regions, which describe the allowed "moves" in the persistence diagram.

▶ **Definition 26** (Forbidden regions). *For an off-diagonal pair $\alpha \in \hat{\mathrm{BD}}(h)$, we say that:*
(1) Forbidden region for $\alpha^\times$ *is defined as*

$$\mathcal{R}_h^{\urcorner}(\alpha) := \bigcup_{\substack{\beta \xrightarrow{\times} \alpha \\ \beta \in \hat{\mathrm{BD}}_n(h)}} [-\infty, h(\beta^\circ)] \times [-\infty, h(\beta^\times)] \;\cup \bigcup_{\substack{\alpha^\times \rightsquigarrow x \\ x \in \mathrm{Crit}(\mathcal{V}_h)}} [-\infty, h(x)] \times [-\infty, h(x)].$$

(2) Forbidden region for α° *is defined as*

$$\mathcal{R}_h^{\llcorner}(\alpha) := \bigcup_{\substack{\beta \xrightarrow{\circ} \alpha \\ \beta \in \hat{\mathrm{BD}}_n(h)}} [h(\beta^\circ), +\infty] \times [h(\beta^\times), +\infty] \;\cup \bigcup_{\substack{y \rightsquigarrow \alpha^\circ \\ y \in \mathrm{Crit}(\mathcal{V}_h)}} [h(y), +\infty] \times [h(y), +\infty].$$

Once we have the notion of forbidden regions, we can define a set of safe transformations, which we call *allowed moves*.

▶ **Definition 27** (Allowed moves). *Let h be a dMf, $\alpha \in \hat{\mathrm{BD}}(h)$ and $c \in \alpha$. A* pre-allowed move *of α is a new dMf h' such that:*
(1) *$\mathcal{V}_h = \mathcal{V}_{h'}$ and for all $x \in \mathrm{Crit}(\mathcal{V}_h) \setminus \{c\}$ we have $h(x) = h'(x)$,*
(2) *If c is a birth cell, then $h'(c) > h(c)$; if c is a death cell, then $h'(c) < h(c)$,*
(3) *h-order and h'-order restricted to $\mathrm{Crit}(\mathcal{V}_h)$ differ by a single transposition at most.*
If a pre-allowed move h' is such that $\mathrm{BD}(h) = \mathrm{BD}(h')$, then we say that h' is an allowed move.

A pre-allowed move pushes the pair α containing c toward the diagonal either by increasing birth or decreasing death without affecting the vector field. A single pre-allowed move bypasses at most one other critical cell. We note that multiple vectors can change their value and position in the h'-order – as long as the gradient structure is preserved. We will use the allowed moves to construct the homotopy bringing a persistent pair to the diagonal.

▶ **Corollary 28.** *If h' is a pre-allowed move for h, then the change in persistence pairing can only result from transpositions of critical cells.*

Proof. The linear homotopy from h to h' may be expressed as a series of transpositions in h-order, given by specific times $t \in [0, 1]$ and h_t-orders. By Theorem 25, the transpositions do not change the vector field, and thus, the diagonal pairs. Therefore, by Proposition 23, the change in persistence pairing can only result from transpositions of critical cells. ◀

Observe that for an X filtered by dMf h, for every cell x and interval $[a, b]$ such that $h(x) \in [a, b]$, we can find $t \in [0, 1]$ such that $h(x) = at + (1 - t)b$. We call it the *linear coefficient of x* on $[a, b]$. We now show that, for a fixed h, one can construct a pre-allowed move that pushes the chosen birth-death pair to the right, and another one that pushes it downward.

▶ **Proposition 29** (Increasing birth – moving right). *Let X be filtered by a dMf h. Let $\alpha \in \hat{\mathrm{BD}}(h)_k$ be an off-diagonal pair, and δ, ξ be real values such that $h(\alpha^\circ) < \delta < \xi < h(\alpha^\times)$, and there is at most one $e \in \mathrm{Crit}(\mathcal{V}_h)$ such that $h(e) \in (h(\alpha^\circ), \delta)$. Additionally, assume that $e \not\rightsquigarrow \alpha^\circ$ and $h^{-1}([\delta, \xi]) = \emptyset$. Define*

$$h'(x) = \begin{cases} t_x\delta + (1-t_x)\xi & \text{when } h(x) \in [h(\alpha^\circ), \xi] \text{ and } x \overset{\mathcal{V}_h}{\rightsquigarrow} \alpha^\circ \text{ and } x \notin \mathrm{Crit}(\mathcal{V}_h) \setminus \{\alpha^\circ\}, \\ h(x) & \text{otherwise,} \end{cases}$$

where t_x is the linear coefficient of x on the interval $[h(\alpha^\circ), \xi]$. Then h' is a pre-allowed move of α with respect to h.

▶ **Proposition 30** (Decreasing death – moving down). *Let X be filtered by dMf h. Let $\alpha \in \hat{\mathrm{BD}}(h)_k$ be an off-diagonal pair, and ξ, δ be real values such that $h(\alpha^\circ) < \xi < \delta < h(\alpha^\times)$, and there is at most one $e \in \mathrm{Crit}(\mathcal{V}_h)$ such that $h(e) \in (\delta, h(\alpha^\times))$. Additionally, assume $\alpha^\times \not\rightsquigarrow e$ and at the same time $h^{-1}([\xi, \delta]) = \emptyset$. Define*

$$h'(x) = \begin{cases} t_x\xi + (1-t_x)\delta & \text{when } h(x) \in [\xi, h(\alpha^\times)] \text{ and } \alpha^\times \overset{\mathcal{V}_h}{\rightsquigarrow} x \text{ and } x \notin \mathrm{Crit}(\mathcal{V}_h) \setminus \{\alpha^\times\}, \\ h(x) & \text{otherwise,} \end{cases}$$

where t_x is the linear coefficient of x on the interval $[\xi, h(\alpha^\times)]$. Then h' is a pre-allowed move of α with respect to h.

Now observe that an allowed move of α does not introduce new forbidden regions.

▶ **Lemma 31.** *Let h' be an allowed move of $\alpha \in \hat{\mathrm{BD}}(h)$. Then $\mathcal{R}^{\urcorner}_{h'}(\alpha) \subset \mathcal{R}^{\urcorner}_{h}(\alpha)$ and $\mathcal{R}^{\llcorner}_{h'}(\alpha) \subset \mathcal{R}^{\llcorner}_{h}(\alpha)$.*

Proof. The statement follows directly from Proposition 22 and the fact that $\mathcal{V}_h = \mathcal{V}_{h'}$, and we are changing the value of only one component of α. ◀

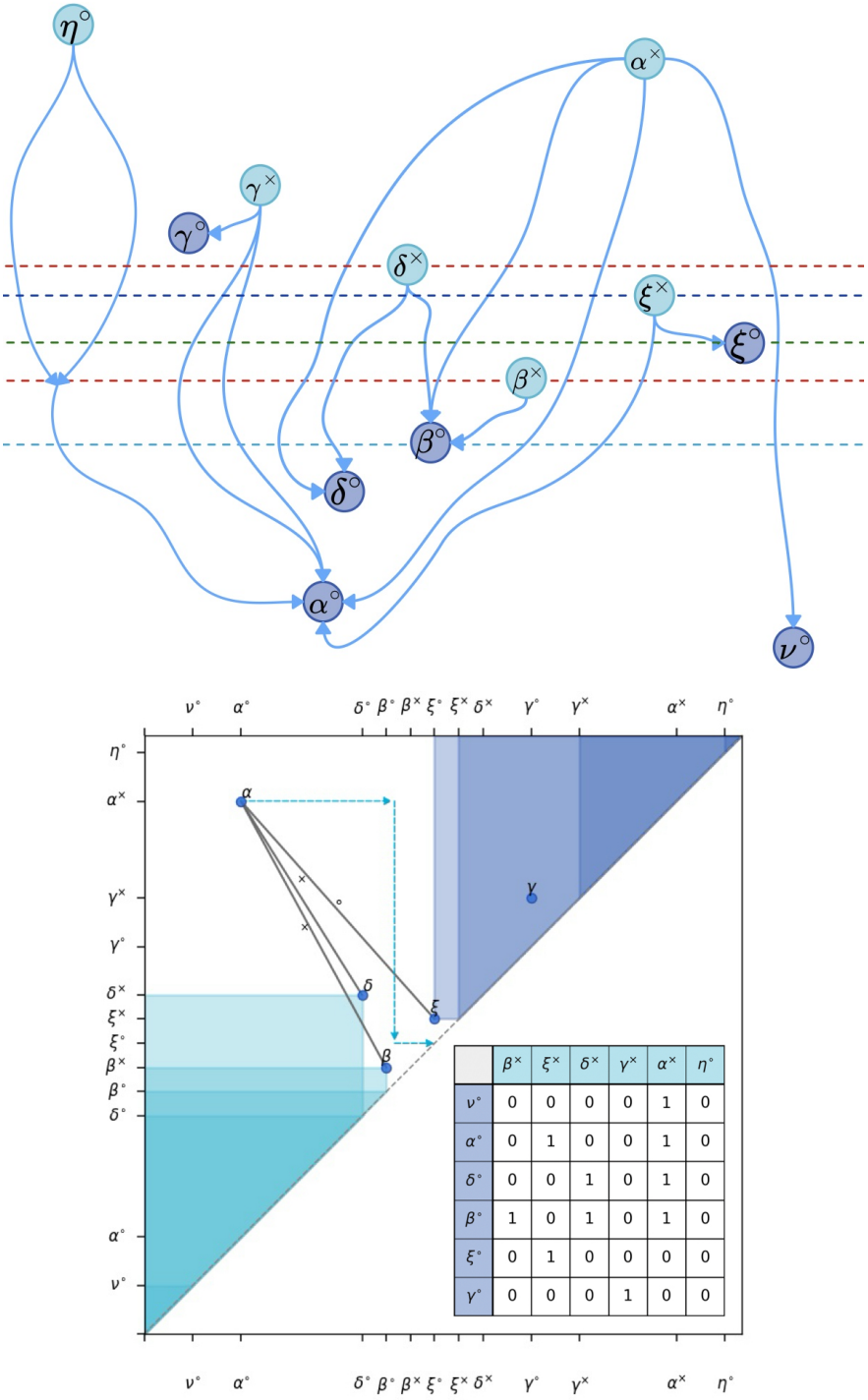
It follows that if we know the initial forbidden regions, we can design a sequence of allowed moves that brings α arbitrarily close to the diagonal.

▶ **Theorem 32.** *Let X be filtered by a dMf h_1 and $\alpha \in \hat{\mathrm{BD}}_k(h)$ be such that $\mathcal{R}^{\llcorner}_{h_1}(\alpha) \cap \mathcal{R}^{\urcorner}_{h_1}(\alpha) = \emptyset$, then there exists a sequence of dMfs $h_1, h_2, \ldots, h_n$ such that h_{i+1} is an allowed move of $\alpha \in \mathrm{BD}(h_i)$, and $h_n(\alpha^\times) - h_n(\alpha^\circ)$ is arbitrarily small.*

Proof. We begin by showing that we are able to construct from h_i an allowed move h_{i+1} such that α° and $\alpha^\times$ are closer in h_{i+1}-order than in h_i-order, where both orders are restricted to the critical cells. Due to Corollary 28, to show that $\mathrm{BD}(h_i) = \mathrm{BD}(h_{i+1})$, we can focus only on transpositions between critical cells. Define x and y as critical cells such that in h_i-order, we have $\alpha^\circ <_{h_i} x <_{h_i} \ldots <_{h_i} y <_{h_i} \alpha^\times$.

If $x \not\rightsquigarrow \alpha^\circ$, then if x is a death cell or $x \overset{\circ}{\not\to} \alpha^\circ$, we use Proposition 29 to construct h_{i+1}, which increases the value of α° with the values of δ and ξ in the proposition larger than $h_i(x)$. Analogously, if $\alpha^\times \not\rightsquigarrow y$, then if y is a birth cell or $y \overset{\times}{\not\to} \alpha^\times$, then use Proposition 30 to construct h_{i+1}, which decreases the value of $\alpha^\times$ to bypass y. From Theorems 19, 20 and 21, we get that these are indeed allowed moves.

If $x \rightsquigarrow \alpha^\circ$ and $\alpha^\times \rightsquigarrow y$, then x generates forbidden regions bounded by a vertical line and y generates forbidden region bounded by a horizontal line. Because $x <_{h_i} y$, they intersect. We get the same argument if $x \rightsquigarrow \alpha^\circ$ and $y \overset{\times}{\to} \alpha^\times$, or $\alpha^\times \rightsquigarrow y$ and $x \overset{\circ}{\to} \alpha^\circ$. If $x \overset{\circ}{\to} \alpha^\circ$ and



	$\beta^\times$	$\xi^\times$	$\delta^\times$	$\gamma^\times$	$\alpha^\times$	η°
ν°	0	0	0	0	1	0
α°	0	1	0	0	1	0
δ°	0	0	1	0	1	0
β°	1	0	1	0	1	0
ξ°	0	1	0	0	0	0
γ°	0	0	0	1	0	0

Figure 3 Top: Schematic picture of $\mathcal{V}_h^k$. Node heights encode values of dMf, critical cells are labeled by Greek letters with superscripts. Several important sublevels are highlighted with dashed lines. Bottom: Boundary matrix and the persistence diagram of the Morse complex induced by $\mathcal{V}_h^k$ with the two kinds of forbidden regions highlighted, and relations involving the birth-death pair α shown as edges. The forbidden regions for $\alpha^\times$ are shown in light blue; those for α°, in darker blue. The dashed arrows illustrate a possible homotopy, which moves the point to the diagonal.

$y \xrightarrow{\times} \alpha^\times$, then due to Corollary 24, there has to exist $(x, \beta^\times), (\gamma^\circ, y) \in \mathrm{BD}_k(h_i)$. It follows from Observation 4 that they are in the bottom-right quadrant of the pair α. Therefore, they generate forbidden $\alpha^\times$ and α° regions, which intersect.

It follows from Lemma 31 that if $\mathcal{R}^{\urcorner}_{h_i}(\alpha) \cap \mathcal{R}^{\llcorner}_{h_i}(\alpha) \neq \emptyset$, then $\mathcal{R}^{\urcorner}_{h_1}(\alpha) \cap \mathcal{R}^{\llcorner}_{h_1}(\alpha) \neq \emptyset$.

Accordingly, we can construct a series of allowed moves, such that h_j is the last one, and α° and $\alpha^\times$ are consecutive in the h_j-order restricted to the critical cells. Then, by Proposition 29, we construct a final pre-allowed move such that the value gap between the cells of α can be made arbitrarily small. Since no critical cell is bypassed during this deformation, the move is allowed. ◀

4.3 Reversing the path

In the previous subsection, we showed that if the forbidden regions of the birth and the death cell of a (reversible) pair α do not intersect, then we can reduce its lifetime arbitrarily close to zero. In particular, we can make it a critical shallow pair. It follows from Theorem 16 that we can safely – that is, without introducing changes in the pairing or in the relations – reverse the path between the components of α. The reversal is the final step of the construction, which corresponds to α entering the diagonal. To make the homotopy fully explicit we construct the final dMf inducing the vector field with reversed path.

▶ **Proposition 33.** *Let $\alpha \in \mathrm{BD}_n(h)$ be a reversible pair such that the unique path between $\alpha^\times$ and α° is ρ. If $h^{-1}([h(\alpha^\circ), h(\alpha^\times)]) = \rho = (\alpha^\circ = x_0, x_1, x_2, \ldots, x_m = \alpha^\times)$, then the function h', defined as*

$$h'(x) = \begin{cases} h(x_{m-2\lfloor i/2 \rfloor}) & \text{when } x \in \rho \text{ and } x = x_i, \\ h(x) & \text{otherwise,} \end{cases}$$

is a dMf which generates $\mathcal{V}^{-\rho}$ and does not change the value of the critical cells other than the components of α.

5 Final algorithm and summary

5.1 Final algorithm

We summarize the entire construction in the form of an algorithm that produces a topological simplification of a given dMf.

Input. A Lefschetz complex X filtered by a dMf h; the combinatorial vector field $\mathcal{V}_h$; the set $\mathrm{BD}(h)$ of birth–death pairs; all homology/cohomology relations among the off-diagonal pairs; and a reversible, k-dimensional birth-death pair α, such that $\mathcal{R}^{\urcorner}_h(\alpha) \cap \mathcal{R}^{\llcorner}_h(\alpha) = \emptyset$.

(1) Following the procedure described in Theorem 32, move α so close to the diagonal that $h^{-1}((h(\alpha^\circ), h(\alpha^\times)) = \rho$, where ρ is a unique path between the components of α. During this process, update the relations between the critical cells of $\mathcal{V}_h$ using Theorems 19, 20.

(2) Reverse the path ρ between $\alpha^\times$ and α° in the vector field, constructing a new dMf as described in Proposition 33.

Output. A Lefschetz complex X filtered by a dMf h'; the combinatorial vector field $\mathcal{V}_{h'}$; the set $\mathrm{BD}(h')$ of birth-death pairs; all homology/cohomology relations among the off-diagonal pairs.

Figure 4 illustrates an example of a topological simplification obtained by this procedure.

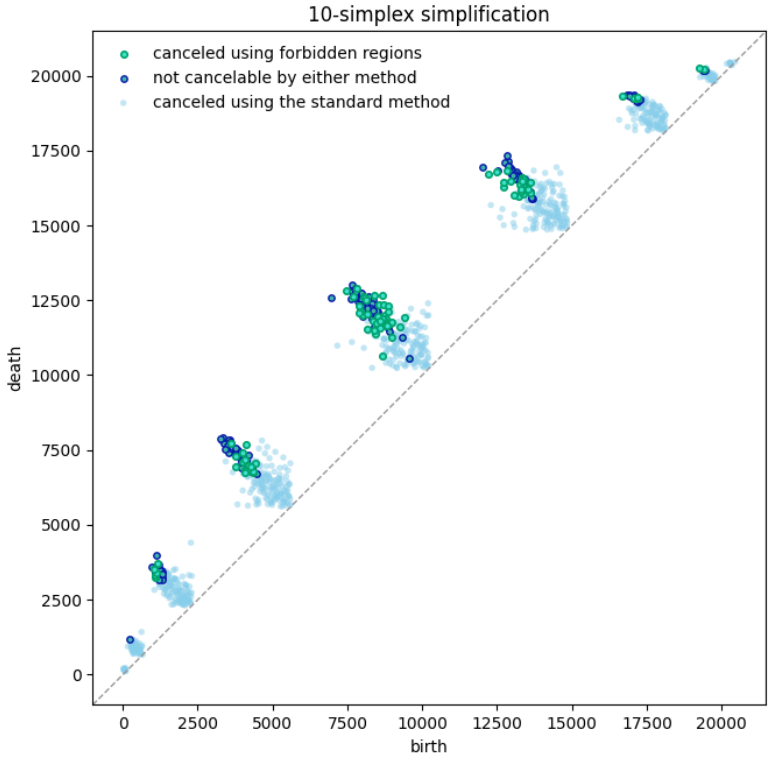


Figure 4 Persistence diagram of the 10-simplex, filtered by a random injective dMf such that every birth-death pair of dimension n is separated from pairs of dimensions $n+1$ and $n-1$. We apply a procedure that first simplifies dMf by the standard method, i.e., path reversing between shallow pairs. When there is no reversible shallow pair left, we continue using the algorithm described in this paper. We made multiple passes canceling any pair that met the algorithm's assumptions. We stopped when there was no reversible pair with a path between forbidden regions. Pairs of different types (canceled by the standard method, canceled using forbidden regions, not cancelable) are denoted by different colors. Figure 5 zooms-in on the pairs in dimension 4.

▶ **Theorem 34.** *The algorithm above returns a topological simplification h' for a dMf h. Moreover, the output of the algorithm contains the updated vector field and homology/cohomology relations for h'.*

Proof. We start by showing that $\hat{\mathrm{BD}}(h') = \hat{\mathrm{BD}}(h) \setminus \{\alpha\}$. Step 1 does not cause any changes in the pairing by Theorem 32. A dMf constructed in Step 2 has the same off-diagonal pairs as the previous one, except α, due to Theorem 16 and Theorem 12.

Proposition 23 and Theorem 16 imply that it suffices to apply the update pattern and check its conditions only during transpositions of critical cells in Step 1. This results in the updated relations among critical cells at the end of the algorithm.

After the update, the new vector field $\mathcal{V}_{h'} = \mathcal{V}^{-\rho}$, where ρ is a unique path. We know the birth-death pairs $\mathrm{BD}(h')$, as well as all relations between critical cells after the application of the update patterns. ◀

The proof of Theorem 34 also proves Theorem 1. For an iterative execution of the algorithm, we can use its output as an input for the next run; one only needs to provide the next eligible pair. Finally, we consider how much the new constructed dMf h' differs from the original one.

▶ **Proposition 35.** *Let X be filtered by a dMf h, and h' be its topological simplification constructed by our algorithm, which removes pair α. Then, the difference between h' and h is bounded by the lifetime of α, that is* $\max_{x \in X} |h(x) - h'(x)| \leq (h(\alpha^{\times}) - h(\alpha^{\circ}))$.

Proof. It follow directly from the fact that in Propositions 29, 30 and 33, we change only the values of the cells between $h(\alpha^{\circ})$ and $h(\alpha^{\times})$, and if the value of the dMf is changed on x, then the resulting value also lies between $h(\alpha^{\circ})$ and $h(\alpha^{\times})$. ◀

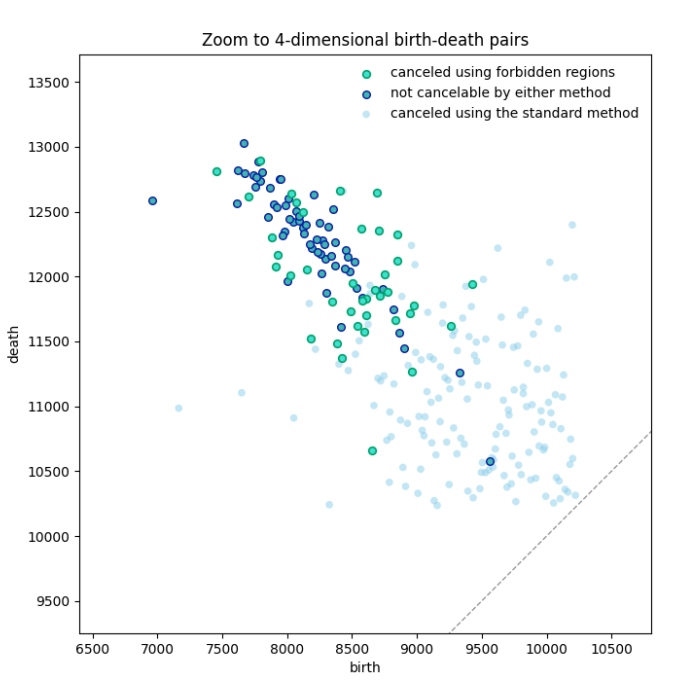


Figure 5 Birth-death pairs in dimension 4 from Figure 4.

5.2 Complexity

We note that checking if a pair α can serve as an input to the algorithm takes $\mathcal{O}(n \log n)$ time, see [23]; there are at most c such pairs to check. The complexity of the algorithm is dominated by the cost of checking if moving pair α past pair β requires updating relations between birth-death pairs, whenever $\beta \nrightarrow \alpha$. Computing $D_{n+1}^{\alpha,\beta}$ can clearly be done in $\mathcal{O}(n^2)$

time; however, in the full version of the paper (see [23]) we show that it can be reduced to $\mathcal{O}(n)$. Therefore, the worst case running time is $\mathcal{O}(c \cdot n)$, where c is the number of birth-death pairs, and n is the number of cells in the complex.

5.3 Summary

We presented a new criterion for removing a fixed birth-death pair. We have also shown that for every pair that satisfies this criterion, it is possible to construct a homotopy, which moves this pair into the diagonal. The paper opens a number of questions.

(1) How does the order of cancellations affect the possibility of canceling the remaining pairs? Is there an optimal order? Can we find a hierarchy of cancellations using this order?
(2) Is the criterion exhaustive? That is, are there other removable pairs that are not captured by the criterion?
(3) Is it possible to weaken the criterion by proper manipulation of the pairs generating the forbidden regions? For example, if forbidden region of α° and forbidden region of $\alpha^\times$ intersect, is it possible to manipulate other cells to clear a path to the diagonal for α, and restore their values after the cancellation?
(4) Is it possible to parallelize the cancellation process? If so, for which pairs?

References

1 Dominique Attali, Marc Glisse, Samuel Hornus, Francis Lazarus, and Dmitriy Morozov. Persistence-sensitive simplification of functions on surfaces in linear time. *Manuscript, presented at TOPOINVIS*, 9:23–24, 2009.

2 Ulrich Bauer. Ripser: efficient computation of Vietoris–Rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423, 2021. doi:10.1007/s41468-021-00071-5.

3 Ulrich Bauer, Carsten Lange, and Max Wardetzky. Optimal topological simplification of discrete functions on surfaces. *Discrete & Computational Geometry*, 47(2):347–377, 2012. doi:10.1007/s00454-011-9350-z.

4 Ulrich Bauer and Fabian Roll. Wrapping Cycles in Delaunay Complexes: Bridging Persistent Homology and Discrete Morse Theory. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:16, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.15.

5 David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. Vines and vineyards by updating persistence in linear time. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry (SoCG'06)*, pages 119–126. ACM, 2006. doi:10.1145/1137856.1137877.

6 Lidija Comic and Leila De Floriani. Cancellation of critical points in 2d and 3d morse and morse-smale complexes. In David Coeurjolly, Isabelle Sivignon, Laure Tougne, and Florent Dupont, editors, *Discrete Geometry for Computer Imagery, 14th IAPR International Conference, DGCI 2008, Lyon, France, April 16-18, 2008. Proceedings*, volume 4992 of *Lecture Notes in Computer Science*, pages 117–128, Berlin, Heidelberg, 2008. Springer. doi:10.1007/978-3-540-79126-3_12.

7 Lidija Comic, Leila De Floriani, Federico Iuricich, and Paola Magillo. Computing a discrete morse gradient from a watershed decomposition. *Comput. Graph.*, 58:43–52, 2016. doi:10.1016/j.cag.2016.05.020.

8 Vin de Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co)homology. *Inverse problems*, 27(12):124003, November 2011. doi:10.1088/0266-5611/27/12/124003.

9 Olaf Delgado-Friedrichs, Vanessa Robins, and Adrian Sheppard. Skeletonization and partitioning of digital images using discrete Morse theory. *IEEE Trans Pattern Anal Mach Intell*, 37(3):654–666, March 2015. doi:10.1109/TPAMI.2014.2346172.

10 Tamal K. Dey, Jiayuan Wang, and Yusu Wang. Improved road network reconstruction using discrete Morse theory. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '17)*, pages 58:1–58:4, New York, NY, USA, 2017. ACM. doi:10.1145/3139958.3140031.

11 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002. doi:10.1007/s00454-002-2885-2.

12 Herbert Edelsbrunner, Michał Lipiński, Marian Mrozek, and Manuel Soriano Trigueros. The poset of cancellations in a filtered complex. arXiv:2311.14364v3, 2024. arXiv:2311.14364.

13 Herbert Edelsbrunner, Michał Lipiński, Marian Mrozek, Manuel Soriano Trigueros, and Fedor Zimin. The depth poset under transpositions in the filter. arXiv:2511.21961, 2025. doi:10.48550/arXiv.2511.21961.

14 Herbert Edelsbrunner, Dmitriy Morozov, and Valerio Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, SCG '06, pages 127–134, New York, NY, USA, 2006. ACM. doi:10.1145/1137856.1137878.

15 Herbert Edelsbrunner and Marian Mrozek. The depth poset of a filtered Lefschetz complex. arXiv:2311.14364v2, 2023. doi:10.48550/arXiv.2311.14364.

16 Riccardo Fellegara, Federico luricich, Leila De Floriani, and Kenneth Weiss. Efficient computation and simplification of discrete Morse decompositions on triangulated terrains. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 223–232, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2666310.2666412.

17 Robin Forman. Combinatorial vector fields and dynamical systems. *Mathematische Zeitschrift*, 228(4):629–681, 1998. doi:10.1007/PL00004638.

18 Robin Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134(1):90–145, 1998. doi:10.1006/aima.1997.1650.

19 Robin Forman. A user's guide to discrete Morse theory. *Séminaire Lotharingien de Combinatoire [electronic only]*, 48:B48c, 35 p., electronic only–B48c, 35 p., electronic only, 2002. URL: http://eudml.org/doc/123837.

20 Rickard Brüel Gabrielsson, Vignesh Ganapathi-Subramanian, Primoz Skraba, and Leonidas J. Guibas. Topology-aware surface reconstruction for point clouds. *Comput. Graph. Forum*, 39(5):197–207, 2020. doi:10.1111/cgf.14079.

21 David Günther, Jan Reininghaus, Hubert Wagner, and Ingrid Hotz. Efficient computation of 3D Morse–Smale complexes and persistent homology using discrete Morse theory. *The Visual Computer*, 28(10):959–969, 2012. doi:10.1007/s00371-012-0726-8.

22 Attila Gyulassy, Peer-Timo Bremer, Bernd Hamann, and Valerio Pascucci. A practical approach to Morse-Smale complex computation: Scalability and generality. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1619–1626, 2008. doi:10.1109/TVCG.2008.110.

23 Jakub Leśkiewicz, Bartosz Furmanek, Michał Lipiński, and Dmitriy Morozov. Topological simplification guided by forbidden regions, 2026. doi:10.48550/arXiv.2603.16416.

24 John Milnor. *Morse Theory*. Princeton University Press, 1963.

25 John Milnor. *Lectures on the h-Cobordism Theorem*. Princeton University Press, 1965.

26 Arnur Nigmetov and Dmitriy Morozov. Topological optimization with big steps. *Discrete & Computational Geometry*, 72:310–344, 2024. doi:10.1007/s00454-023-00613-x.

27 Vanessa Robins, Peter John Wood, and Adrian P. Sheppard. Theory and algorithms for constructing discrete Morse complexes from grayscale digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1646–1658, 2011. doi:10.1109/TPAMI.2011.95.

The Complete 10-Tetrahedra Census of Orientable Cusped Hyperbolic 3-Manifolds

Shana Yunsheng Li
Dept. of Math., University of Illinois Urbana-Champaign, IL, USA

Abstract

We extend the complete census of orientable cusped hyperbolic 3-manifolds to 10 tetrahedra, giving the next 150,730 manifolds and their 496,638 minimal ideal triangulations. As applications, we find the precisely 439,898 exceptional Dehn fillings on them, revealing the next 1,849 simplest hyperbolic knot exteriors in S^3. We also give the simplest example of an orientable cusped hyperbolic 3-manifold containing a closed totally geodesic surface.

2012 ACM Subject Classification Mathematics of computing → Geometric topology

Keywords and phrases hyperbolic manifolds, 3-manifolds, triangulation, census, tabulation, exact computation, computational topology, low-dimensional topology

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.73

Related Version *Full Version*: `https://arxiv.org/abs/2512.02142` [33]

Supplementary Material *Dataset*: `https://doi.org/10.7910/DVN/BFEXZA` [34]

Funding This material is based upon work supported by the National Science Foundation under Grant No. DMS-2424139 while the author was in residence at the Simons Laufer Mathematical Sciences Institute in Berkeley, California, during the Spring 2026 semester. The author was also supported by the National Science Foundation under Grant No. DMS-2303572 during the same semester.

Acknowledgements The author wishes to thank Nathan Dunfield for his generous guidance during numerous discussions, and the Keeling cluster maintenance team at UIUC for providing the computational resources that made this project possible.

1 Introduction

The enumeration of examples has been of fundamental importance in the study of low-dimensional topology ever since Tait gave the first knot table in 1884 [41]. The censuses of knots have been extensively explored over the years, all the way to approximately the first 2 billion knots [28, 8, 43]. However, the censuses of orientable cusped hyperbolic 3-manifolds, initiated by Hildebrand and Weeks [25], have seen no progress after the 44,250 manifolds whose minimal ideal triangulations consist of 9 tetrahedra were given by Burton in 2014 [7]. In this paper, we extend the census by giving the 150,730 manifolds whose minimal ideal triangulations consist of 10 tetrahedra. Table 1 summarizes the timeline of the development of censuses of orientable cusped hyperbolic 3-manifolds.

By hyperbolic 3-manifolds, we mean 3-manifolds with Riemannian metrics of constant curvature -1 that are complete as metric spaces; in particular their geodesics can be extended indefinitely in length. A hyperbolic 3-manifold is *cusped* if it has a topological end that is of finite volume. An orientable cusped finite-volume hyperbolic 3-manifold is homeomorphic to the interior of a compact 3-manifold whose boundary is a non-empty union of tori. Our main result is the following:

▶ **Theorem 1.** *There are precisely* 150,730 *orientable cusped hyperbolic* 3*-manifolds whose minimal ideal triangulations consist of* 10 *tetrahedra. Moreover, there are precisely a total of* 496,638 *minimal ideal triangulations of these manifolds.*

© Shana Yunsheng Li;
licensed under Creative Commons License CC-BY 4.0

Table 1 Timeline of censuses of orientable cusped hyperbolic 3-manifolds.

Tetrahedra	Name of manifolds	Year	Contributor(s)
2 - 5	m003 ∼ m412	1989	Hildebrand & Weeks [25]
6 7	s000 ∼ s961 v0000 ∼ v3551	1999	Callahan, Hildebrand & Weeks [11]
8	t00000 ∼ t12845	2010	Thistlethwaite [42]
9	o9_00000 ∼ o9_44249	2014	Burton [7]
10	o10_000000 ∼ o10_150729	2025	L.

Table 5 in [33, Appendix A.2] gives the counts of manifolds in Theorem 1 by cusp number.

We highlight here that in order to obtain Theorem 1, we applied a new core technique, the verified canonical triangulations. We anticipate that a mixture of this technique with traditional ones would suffice for further extension of the census, provided successful generation of the candidates; see Sections 3.1 and 3.5 for more detailed discussions.

Throughout this paper, we assume triangulations consist of finitely many tetrahedra. If a cusped hyperbolic 3-manifold admits an ideal triangulation, each tetrahedron has finite volume [15, §2.5], hence the volume of the manifold is finite. Conversely, any cusped finite-volume hyperbolic 3-manifold has an ideal triangulation given by a refinement of the Epstein–Penner cellulation [37, §5]. Therefore, for cusped hyperbolic 3-manifolds, being finite-volume is equivalent to admitting ideal triangulations, hence we drop the term finite-volume when talking about cusped hyperbolic 3-manifolds admitting ideal triangulations.

In general, it is unknown whether all cusped finite-volume hyperbolic 3-manifolds admit geometric triangulations in the sense of Section 2.1 below [36]. However, when certifying hyperbolicity, `SnapPy` returns true if and only if it finds a geometric triangulation. Therefore an immediate corollary of this paper is that

▶ **Corollary 2.** *All cusped hyperbolic* 3*-manifolds ideally triangulable by at most* 10 *tetrahedra admit geometric triangulations.*

Note that nonorientable manifolds are included here since they are included in Theorem 7. For orientable manifolds, the above result can be strengthened to Proposition 11.

We briefly review the needed terminologies in Section 2, then give detailed discussions in Section 3 for the creation of the 10-tetrahedra census, after which we give some applications of the extended census in Section 4. All corresponding code and data are available at [34].

2 Preliminaries

2.1 Triangulations

A *(generalised) triangulation* $\mathcal{T}$ is a disjoint union of finitely many tetrahedra, with some or all faces identified in pairs via simplicial maps, and we avoid identifications where an edge is identified with itself in reverse. Note that $\mathcal{T}$ is not necessarily a simplicial complex, as identifications of two faces of the same tetrahedron are allowed.

Quotienting by the identifications, $\mathcal{T}$ can be seen as a topological space, along with a cellular structure induced by the tetrahedra. A *vertex* in $\mathcal{T}$ is a 0-cell in this cellular structure, and an *edge* is a 1-cell. The *link* of a vertex in $\mathcal{T}$ is the frontier of a small regular neighborhood of that vertex. A vertex is *internal* if its link is a sphere, and is *ideal* if its link is some other closed surface. For an edge e in $\mathcal{T}$, its *degree* is the cardinality of the set

$$\mathcal{S}_e \coloneqq \bigcup_{t \in \{\text{3-cells of } \mathcal{T}\}} \{\text{(nonempty) connected components of } \mathcal{N}_e \cap t\},$$

where $\mathcal{N}_e$ is a small regular neighborhood of the midpoint of e.

▶ **Definition 3.** *Let M be the interior of a compact 3-manifold with boundary. A triangulation $\mathcal{T}$ is an ideal triangulation of M if it is homeomorphic to M after removing its ideal vertices, and is minimal if no ideal triangulation of M with fewer tetrahedra exists.*

It follows that every vertex of an ideal triangulation is either internal or ideal. Note that an ideal triangulation itself is not a manifold and has at least one ideal vertex.

If M is a cusped hyperbolic 3-manifold, an ideal triangulation $\mathcal{T}$ of M is *geometric* if all vertices are ideal, all edges are geodesics, all 2-cells are parts of geodesic planes, and no flat or negatively oriented tetrahedron (i.e. with zero or negative volume) exists.

The following results were proven in Section 3 of [7]:

▶ **Lemma 4** (Burton). *Let $\mathcal{T}$ be a minimal ideal triangulation of a cusped hyperbolic 3-manifold. Then $\mathcal{T}$ contains no internal vertices.*

▶ **Lemma 5** (Burton). *Let $\mathcal{T}$ be a minimal ideal triangulation of a cusped hyperbolic 3-manifold. Then $\mathcal{T}$ has no edges of degree 1 or 2, and $\mathcal{T}$ has no edges of degree 3 that are contained in three distinct tetrahedra.*

2.2 Normal surfaces

In a tetrahedron, an arc α properly embedded in one of its faces is *normal* if the endpoints of α are in the interior of distinct edges. A properly embedded disk is a *triangle* if its boundary consists of three normal arcs in distinct faces, and is a *square* if its boundary consists of four normal arcs in distinct faces.

Let $\mathcal{T}$ be a triangulation. A properly embedded compact surface in $\mathcal{T}$ is *normal* if its intersection with every tetrahedron in $\mathcal{T}$ (with the identifications undone) is a properly embedded surface whose connected components consist of only triangles and squares.

An isotopy of $\mathcal{T}$ is normal if it preserves the cellular structure of $\mathcal{T}$. Up to normal isotopy, there are 4 triangles and 3 squares in a tetrahedron. Multiple copies of triangles or squares in the same normal isotopy class may appear in the intersection of a normal surface and a tetrahedron. It follows that a normal surface in a triangulation of n tetrahedra is uniquely determined by a vector in $\mathbb{N}^{7n}$ up to normal isotopy, where $\mathbb{N} \coloneqq \{k \in \mathbb{Z} \mid k \geq 0\}$.

The subset $\mathcal{N} \coloneqq \{\boldsymbol{v} \in \mathbb{N}^{7n} \mid \boldsymbol{v} \text{ represents a normal surface}\}$ consists of integral vectors in a certain non-convex subset of a polyhedral cone in $\mathbb{R}^{7n}$ [29]. The *standard vertex normal surfaces* correspond to vertices of a cross-section of this polyhedral cone, and can be viewed as forming a finite basis for $\mathcal{N}$.

The following result was proven in Section 3 of [7]:

▶ **Lemma 6** (Burton). *Let $\mathcal{T}$ be a minimal ideal triangulation of a cusped hyperbolic 3-manifold. Then no standard vertex normal surface in $\mathcal{T}$*

- *has positive Euler characteristic; or*
- *has zero Euler characteristic, is not a vertex link, and in addition*
  - *is one-sided; or*
  - *is two-sided, and if we cut $\mathcal{T}$ open along it then no component of the resulting space is a solid torus or solid Klein bottle.*

3 Creating the 10-tetrahedra census

3.1 Overview

To build a census (of knots, manifolds, etc), the procedure generally follows a 3-step outline:

(i) Generate all candidates for the census.

(ii) Decide the eligibility of each candidate.

(iii) Group eligible candidates into isomorphism classes and certify that the isomorphism classes are all distinct.

As long as the candidate generation is performed exhaustively and the eligibility for each candidate is decided correctly, the census obtained is guaranteed to contain all wanted objects. With step (iii), the resulting census will contain no duplicates. We say that a census is *complete* if it contains all wanted objects without any duplicated ones.

For censuses of cusped hyperbolic 3-manifolds, the candidates will be ideal triangulations consisting of a given number of tetrahedra, and the eligibility conditions will be minimality (of triangulations) and hyperbolicity (of manifolds the triangulations represent). For certifying distinctness, algebraic invariants were widely applied for both the census of hyperbolic 3-manifolds [7] and the census of knots [28, 8, 43]. However, it is unknown whether algebraic methods will indeed separate all given isometry classes within a given scope, consequently they cannot be used to certify if two isometry classes are indeed the same, and other methods will be required for that purpose. Here we propose an alternative approach for cusped hyperbolic 3-manifolds, using a geometric invariant, the *canonical triangulation*, built upon the Epstein–Penner canonical cellulation [21] by Weeks [44], whose rigorous computation was implemented by Goerner [24]. An introduction to these can be found in [22], and we review them briefly in Section 3.3.1. The canonical triangulation, when computed rigorously, is a complete invariant of cusped hyperbolic 3-manifolds, and we applied it to deduplicate a majority of triangulations in the 10-tetrahedra census in Section 3.3.3.

Our approaches for steps (i) and (ii) largely overlap with those in [7]. For completeness, we briefly describe them in Section 3.2, and kindly refer the reader to [7] for more details. For step (iii), detailed discussions will be given in Section 3.3.

3.2 Candidate generation and eligibility tests

Let us start by looking at Table 2. The steps proceed from top to bottom. Each row consists of the name of the step, the number of remaining candidates whose hyperbolicity is unknown, the numbers of candidate discarded within that step due to ineligibility, and the number of candidates kept to be dealt with later in Section 3.3.3.

For candidate generation, we used `Regina`'s utility command `tricensus` to generate all ideal triangulations consisting of 10 tetrahedra except those that are obviously nonminimal or non-hyperbolic. More precisely, we obtained:

▶ **Theorem 7.** *Up to relabelling, there are* 8,373,308 *ideal triangulations with* 10 *tetrahedra, in which every vertex is ideal, there are no edges of degree* 1 *or* 2*, and there are no edges of degree* 3 *that belong to three distinct tetrahedra.*

Due to Lemmas 4 and 5, the triangulations described in Theorem 7 contain all minimal ideal triangulations of cusped hyperbolic 3-manifolds whose minimal ideal triangulations consist of 10 tetrahedra. Note that triangulations representing nonorientable manifolds are also included and carried through the tests, though we will discard them in Section 3.3.

Table 2 The steps of filtering candidates for the 10-tetrahedra census.

Step	Candidates	Discarded	Kept
Candidate generation	8,373,308	0	0
Greedy nonminimality	1,946,782	6,426,526	0
Certify hyperbolicity ($r = 1$)	1,072,874	0	873,908
Exhaustive nonminimality ($h = 2$)	698,650	374,224	0
Special surfaces	33,807	664,843	0
Certify hyperbolicity ($r = 60$)	3,269	0	30,538
Exhaustive nonminimality ($h = 5$)	12	3,257	0
Exhaustive nonminimality ($h = 6$)	6	6	0
Certify hyperbolicity ($r = 1000$)	0	0	6
Total	8,373,308	7,468,856	904,452

With the candidates generated, we proceed to determine their eligibilities. For an ideal triangulation to remain in the census, it must be both minimal and representing a cusped hyperbolic 3-manifold. Hence an ideal triangulation is discarded immediately if it is certified to be nonminimal or representing a non-hyperbolic 3-manifold. If an ideal triangulation is certified to represent a cusped hyperbolic 3-manifold, we temporarily keep it, and decide whether it is minimal by checking if it represents a manifold that has already appeared in censuses with fewer tetrahedra, which we discuss in Section 3.3.

Four tests were used in Table 2, which we describe below:

Greedy nonminimality: we use `Regina`'s highly efficient simplification algorithm [5] to try converting each candidate into an ideal triangulation with fewer tetrahedra, while representing the same 3-manifold. If such a triangulation with fewer tetrahedra is found, then the candidate is nonminimal.

Exhaustive nonminimality: given an integer parameter $h > 0$, for each candidate with n tetrahedra ($n = 10$ in our settings), enumerate all triangulations that can be reached via 2-3 and 3-2 Pachner moves without ever exceeding $n + h$ tetrahedra. If a triangulation with fewer than n tetrahedra is found during the process, the candidate is nonminimal.

Special surfaces: for each candidate $\mathcal{T}$, enumerate all standard vertex normal surfaces in it, and check if any of them

- has positive Euler characteristic; or
- has zero Euler characteristic, is not a vertex link, and in addition
  - is one-sided; or
  - is two-sided, and if we cut $\mathcal{T}$ open along it then no component of the resulting space is a solid torus or solid Klein bottle.

If any such surface is found, then by Lemma 6, $\mathcal{T}$ is either nonminimal or does not represent a cusped hyperbolic 3-manifold.

Certify hyperbolicity: given an integer parameter $r > 0$, for each candidate $\mathcal{T}$, we randomize it by doing random Pachner moves, and ask `SnapPy` to try finding a complete hyperbolic structure on $\mathcal{T}$, which uses an algorithm similar to HIKMOT's [27], where affirmative answers are rigorous [24]. If the answer is not affirmative, we randomize $\mathcal{T}$ and try again, until an affirmative answer is achieved or the number of attempts reaches r.

We summarize the tests along with their core functions in Table 4 in [33, Appendix A.1].

As is presented in Table 2, for each candidate, an affirmative answer was given by one of the above four tests. A candidate was kept if and only if the affirmative answer was given by the certify hyperbolicity test, and was discarded otherwise. As such, we obtained 904,452 triangulations certified to represent cusped hyperbolic 3-manifolds at the bottom of Table 2, with all others discarded.

3.3 Grouping and Separating

In this section, we discuss the procedure for step (iii) described in Section 3.1. A central tool for it, in both [7] and our method, is the *canonical triangulation* of cusped finite-volume hyperbolic 3-manifolds.

3.3.1 Canonical triangulation

We review the definition and computation of canonical triangulations very briefly, and refer the reader to [22] for more details.

Consider the $(3+1)$-Minkowski space $\mathbb{E}^{3,1} \coloneqq \mathbb{R}^4$ whose inner product $\langle \cdot, \cdot \rangle$ is defined by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \coloneqq \sum_{i=0}^{2} x_i y_i - x_3 y_3$. The hyperboloid $S^+ \coloneqq \{\boldsymbol{x} \in \mathbb{E}^{3,1} \mid x_3 > 0, \quad \langle \boldsymbol{x}, \boldsymbol{x} \rangle = -1\}$ is isometric to $\mathbb{H}^3$. Given a cusped hyperbolic 3-manifold M, for each cusp of M we choose a horotorus cusp neighborhood of the same volume; each neighborhood is bounded by a torus, which lifts to horospheres in $\mathbb{H}^3 \cong S^+$. For each horosphere $\mathcal{H} \subset S^+$, its normal vector $\boldsymbol{v}$ in $\mathbb{E}^{3,1}$ satisfies $\langle \boldsymbol{v}, \boldsymbol{v} \rangle = 0$, and we normalize it so that $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = -1$ for any $\boldsymbol{w} \in \mathcal{H}$. Therefore we obtain a family $\{\boldsymbol{v}\}_M$ of such vectors associated to a cusped hyperbolic 3-manifold M.

▶ **Definition 8.** *The Epstein–Penner cellulation of M is given by the radial projection of the polygonal faces of the convex hull of $\{\boldsymbol{v}\}_M$ onto S^+.*

The Epstein–Penner cellulation is invariant under the choice of the volume of cusp neighborhoods, hence it is an invariant of cusped hyperbolic 3-manifolds [21]. It is generically a triangulation, but not always so. Given a polygonal cellulation of a 3-manifold, one can subdivide it into the triangulation induced by the following refinement of its 1-skeleton:

- add a vertex in the interior of each 2-cell and 3-cell;
- for each 2-cell or 3-cell e, let v be the vertex added to its interior:
  - for each vertex u originally in e, add an edge connecting v and u;
  - if e is a 3-cell, for each 2-cell e_2 on the boundary of e, add an edge connecting v and the vertex added to the interior of e_2.

▶ **Definition 9.** *The canonical triangulation of M is the Epstein–Penner cellulation of M if the latter is a triangulation, and the subdivision of the Epstein–Penner cellulation described above otherwise.*

Weeks established a heuristic procedure which, given a geometric triangulation, computes the Epstein–Penner cellulation of the corresponding cusped hyperbolic 3-manifold via local modifications, doing 2-3 and 3-2 Pachner moves wherever tetrahedra meet concavely at faces or edges [44]. The canonical triangulation can then be easily computed from the Epstein–Penner cellulation. The methods for both were implemented in `SnapPea` in the 1990s, and later inherited by its successor `SnapPy` [13]. In their original implementations, the hyperbolic structures used were only recorded numerically, hence accuracy issues may come into play, which we discuss in the following two sections.

3.3.2 Burton's method: algebraic certification

Let us briefly review the method Burton used in [7], which can be summarized as:

1. For each ideal triangulation, compute its canonical triangulation numerically using `SnapPea`, and partition them into classes by the canonical triangulations computed.
2. For triangulations in each class, perform an exhaustive search through 2-3 and 3-2 Pachner moves, attempting to identify classes that should have been merged in step 1, but were not due to numerical accuracy issues, thus refining the partitioning.
3. For each class, compute the list $\{\text{orientability}, H_1 = \pi_1^{\text{ab}}, S_2^{\text{ab}}(\pi_1), \dots, S_k^{\text{ab}}(\pi_1)\}$ of algebraic invariants to certify that all groups are distinct, increasing k if necessary, where $S_i^{\text{ab}}(G)$ is the multi-set of abelianisations of index i subgroups of G, one subgroup for each conjugacy class. For the 9-tetrahedra census, $k = 11$ was required and sufficed.

Note that to confirm the completeness of previous censuses, Burton dealt with all candidates with no more than 9 tetrahedra at once, hence triangulations that were grouped with triangulations with fewer tetrahedra were discarded immediately due to nonminimality.

There are several aspects where numerical computation could cause issues in step 1:

- A quantity *tilt* is used to characterize whether tetrahedra meet concavely, parallelly or convexly, each corresponding to the tilt being positive, zero, or negative respectively [44, 40]. When the absolute value of tilt is small, the numerical computation may falsely recognize the sign of tilt, resulting in an incorrect canonical triangulation.
- Flat tetrahedra being falsely recognized as non-flat, which would then be incorrectly dealt with when subdividing Epstein–Penner triangulations into canonical triangulations.

When the above issues arise in step 1, `SnapPea`/`SnapPy` fails to find the correct canonical triangulation and arrives at an incorrect one, while still representing the same manifold as the input triangulation. Hence steps 2 and 3 were necessary to spot and resolve these miscomputed cases. However, steps 2 and 3 can be very expensive to perform, and require back-and-forth adjustments of parameters. Furthermore, although successful for censuses with at most 9 tetrahedra, it is unknown whether step 3 will separate all cases within a computable scope, or even if it will separate all of them in theory [23].

▶ Remark 10. We remark here that the subdividing algorithm Weeks implemented in `SnapPea`/`SnapPy` does not work for refinements of Epstein–Penner cellulations containing flat tetrahedra. This led to the miscomputed example x101 found by Burton [6] and mentioned in [22, Remark 3.6]. The cause of the miscomputation was not located at the time of [22]. During the course of this project, the author found another such example, o9_43806 and located the said cause. This bug has been fixed in `SnapPy 3.3` by forcing it to find a refinement that does not contain flat tetrahedra before proceeding to subdividing it. After the fix, the numerical computation gives correct results for all minimal triangulations in the orientable cusped censuses of at most 10 tetrahedra, which we confirmed using the verified computation explained in Section 3.3.3 (as long as successfully computed).

3.3.3 Our method: verified computation

Instead of making up for the mistakes caused by numerical computation of canonical triangulations using other methods, we enhance the computation itself to avoid the mistakes directly. With the accuracy issues described in the previous section, this requires us to:

(a) given two supposedly different quantities, rigorously tell which one is larger;

(b) given two supposedly equal quantities, prove that they are equal.

For (a), we use interval arithmetics, which can be thought of as computation with controlled errors. If two quantities are different, with sufficiently small error the lower bound of one will be strictly larger than the upper bound of another, revealing the larger quantity. Similar approach was also used in HIKMOT's rigorous certification of hyperbolicity [27].

For (b), we use the LLL algorithm [32] to symbolically solve the trace field of hyperbolic structure, representing it as a number field. We then express the quantities as elements in the number field, hence they are equal if and only if their difference is the zero element.

The above enhanced computation was implemented in `SnapPy` by M. Goerner [24], and is called verified computation. We refer the reader to [20, 22] for more detailed discussions.

Canonical triangulations obtained with verified computation are thus provably correct. Let us call a supposedly canonical triangulation obtained by `SnapPy` a *verified canonical triangulation* if it was obtained with verified computation, and an *unverified canonical triangulation* if the computation was performed purely numerically.

Let $\{\mathcal{T}\}$ be the set of ideal triangulations representing cusped hyperbolic 3-manifolds and $\{\mathcal{M}\}$ be the set of cusped finite-volume hyperbolic 3-manifolds. Let $\iota\colon \{\mathcal{T}\} \to \{\mathcal{M}\}$ be the function that maps an ideal triangulation to the manifold it represents. The unverified canonical triangulation can thus be seen as a multi-valued function $c_u\colon \{\mathcal{M}\} \to \{\mathcal{T}\}$ which is a right inverse of ι; in other words, the following diagram commutes:

$$\begin{array}{ccc} \{\mathcal{T}\} & \xrightarrow{\ \iota\ } & \{\mathcal{M}\} \\ & {\scriptstyle c_u}\nwarrow \quad \uparrow{\scriptstyle 1_{\{\mathcal{M}\}}} & \\ & \{\mathcal{M}\} & \end{array} \tag{1}$$

In particular, c_u is injective. The verified canonical triangulation is then a branch of c_u, a single-valued function $c_v\colon \{\mathcal{M}\} \to \{\mathcal{T}\}$. Since c_u is injective, c_v is also injective, hence restricts to a bijection between $\{\mathcal{M}\}$ and its image. Therefore the verified canonical triangulation is a complete invariant of cusped finite-volume hyperbolic 3-manifolds.

Although the definition of canonical triangulation does not require the manifold to be orientable, the current implementation of verified computation for canonical triangulations in `SnapPy` does not work for nonorientable ones with multiple cusps. Hence we discard the nonorientable candidates and deal with the remaining 608,918 orientable ones.

We successfully computed the verified canonical triangulations for all except 16 of the 608,918 orientable candidates. The 16 cases failed due to their trace fields being too complicated; we remark that this very nature intrinsically separates them from the others, though further investigation is still required to prove it. By comparing the results with the verified canonical triangulations of all manifolds in censuses of at most 9 tetrahedra, we obtained 496,622 minimal triangulations of 10 tetrahedra, which, grouped by their verified canonical triangulations, give 150,724 distinct manifolds. For the 16 candidates whose verified canonical triangulations were not computed, we computed the first 60 digits of their hyperbolic volumes using interval arithmetics [24] and their first homology groups, which separated them from all manifolds in censuses up to 9 tetrahedra plus the 150,724 ones, and grouped them into 6 classes; within each class, the triangulations were confirmed to all represent the same manifold as their unverified canonical triangulations are the same.

Therefore, we conclude that there are exactly 150,730 isometry classes in the 10-tetrahedra census, containing a total of 496,638 minimal triangulations, obtaining Theorem 1.

3.4 Finalizing the census

For each of the 150,730 isometry classes obtained in Section 3.3, we select a representative triangulation by the following procedure:

(1) If the verified canonical triangulation is in the isometry class, let it be the representative.
(2) Otherwise, consider the subset of triangulations that are geometric, which turns out to be never empty. If there is only one triangulation in the class, let it be the representative.
(3) If there are multiple triangulations in that subset, let the triangulation with the most self-isomorphisms, i.e. cellular homeomorphisms onto itself, be the representative.
(4) If there are multiple triangulations with the most self-isomorphisms, choose the one with the "fattest" tetrahedra, i.e. the one with the largest minimum imaginary part of tetrahedron shape parameters.
(5) If there are still multiple options left, choose the one with the smallest triangulation signature (as defined in [4]) under the lexicographical order.

With the representative triangulations chosen as above, we set their peripheral curves to be the shortest. We then ordered them by volume, named them as displayed in Table 1, saved and published the data in [34]. The data now comes automatically with `SnapPy 3.3`.

Note that verified canonical triangulations, by construction, are always geometric [22]. Hence due to the geometric requirement in step (2), we obtained

▶ **Proposition 11.** *Every cusped hyperbolic* 3*-manifold in the* 10*-tetrahedra census admits a minimal triangulation that is geometric.*

We remark that the above proposition also holds for all (orientable) censuses with fewer tetrahedra, which we checked using Burton's data from [7].

3.5 Obstructions on further extending the census

As is discussed in Section 3.1, there are essentially three steps for creating a complete census. For the census of (orientable) cusped hyperbolic 3-manifolds, step (ii) should not be a problem, as there are several other tests proposed in [7], but were not used when creating the 10-tetrahedra census. For step (iii), the verified computation has a chance of failing only when the LLL algorithm is invoked, which happens only when a tilt that is extremely close to zero is encountered during computation. This essentially implies that the Epstein–Penner cellulation is not a triangulation, which is the case for only 986 triangulations out of the 608918 orientable candidates mentioned in Section 3.3.3, a frequency less than 0.2%. Therefore, it is promising that a combination of verified canonical triangulations with algebraic methods would suffice in the foreseeable future, where the former separates most manifolds, and the latter deals with the rest.

The main obstruction thus lives in step (i), where an exponential growth rate of the computational power is needed, with a much larger base compared to tabulating knots [39]. If one were to include nonorientable candidates during generation (as we did in Theorem 7), even though the algorithm established in [7] is already extremely efficient, it still takes approximately 2 years to generate all candidates with 10 tetrahedra in a single thread of 2.6 GHz, about 50 times longer than to generate all candidates with 9 tetrahedra; we accelerated the process by parallelizing it in 300 threads. Thus, assuming the same growth rate, it will take approximately 100 years to generate all candidates with 11 tetrahedra with a single thread of 2.6 GHz, and hence 4 months in 300 threads.

The time consumption and growth rate can both be lowered by excluding nonorientable candidates during generation, as half of identifications of each face will be excluded from the beginning. It takes 2 months to enumerate all orientable candidates with 10 tetrahedra in a

single thread, about 25 times longer than for 9 tetrahedra. We successfully generated all orientable candidates with 11 tetrahedra, which took about 6 years in a single thread, giving Theorem 12. We have now completed the 11-tetrahedra orientable census, with a preliminary `SnapPy` module available at [35], and full details to appear in a future publication.

▶ **Theorem 12.** *Up to relabelling, there are* 27,794,289 *ideal triangulations of orientable* 3*-manifolds with* 11 *tetrahedra, in which every vertex is ideal, there are no edges of degree* 1 *or* 2*, and there are no edges of degree* 3 *that belong to three distinct tetrahedra.*

4 Applications

A substantial amount of follow-up work can be done with the extended census. We describe several of them in this section.

4.1 Extending the census of exceptional Dehn fillings

Let $\overline{M}$ be a compact orientable 3-manifold whose interior M is a 1-cusped hyperbolic 3-manifold; in particular $\partial\overline{M}$ is a torus, and M has finite volume. Dehn fillings on M are characterized by slopes on $\partial\overline{M}$, which are unoriented isotopy classes of simple closed curves on the torus. For a slope α, we denote by $M(\alpha)$ the closed manifold obtained by Dehn filling on M so that α bounds a disk in the attached solid torus $D^2 \times S^1$.

▶ **Definition 13.** *A slope α is exceptional if $M(\alpha)$ is not hyperbolic.*

The census of exceptional Dehn fillings will consist of pairs (M, α), where α is exceptional.

By deleting a cusp neighborhood N of M, we obtain a compact 3-manifold $M \setminus N$ with torus boundary which is isotopic to $\overline{M}$. Slopes on $\partial\overline{M}$ can thus be carried by the isotopy onto $\partial(M \setminus N) \subset M$. By taking N to be maximal, we define the length of a slope to be the minimum length of all curves in its isotopy class on $\partial(M \setminus N)$.

Agol [1] and Lackenby [31] independently proved that lengths of exceptional slopes cannot exceed 6. It follows that there are only finitely many exceptional fillings on a 1-cusped finite-volume hyperbolic 3-manifold. Therefore it is possible to give complete lists of exceptional Dehn fillings on 1-cusped hyperbolic 3-manifolds in the censuses. For censuses of at most 9 tetrahedra, this has been done by Dunfield [17]. For the 10-tetrahedra census, we obtained:

▶ **Theorem 14.** *There are precisely* 439,898 *exceptional Dehn fillings on orientable* 1*-cusped hyperbolic* 3*-manifolds in the* 10*-tetrahedra census.*

Table 3 outlines the procedure for obtaining Theorem 14. There are 144,016 1-cusped hyperbolic 3-manifolds in the 10-tetrahedra census. For each of them, we enumerated all slopes with length no more than 6, obtaining a total of 800,447 candidates for the exceptional census. As no duplicate slopes were enumerated for a given manifold, all candidates are distinct, and we only need to decide the hyperbolicity for each of them.

For each candidate (M, α), we consider its filled manifold $M(\alpha)$. Several rounds of certifying the hyperbolicity of $M(\alpha)$ with `SnapPy` reduced the remaining list by nearly a half. For Regina census lookup, we cut $M(\alpha)$ into connected sum summands and used `regina.Census.lookup` to try obtaining a name classifying it, which then tells us whether $M(\alpha)$ is hyperbolic or not. For those whose names were not obtained, we check whether they have fundamental normal surfaces which are essential S^2 or tori; if they do, they are not hyperbolic. Only 11 candidates remain undetermined after this, and their filled manifolds were all identified as m135(1,3) by `SnapPy`. m135(1,3) is isometric to m412(1,-3)(-3,1) whose triangulation in `SnapPy` is certified to be geometric, hence m135(1,3) is hyperbolic.

Table 3 The steps for creating the 10-tetrahedra exceptional census.

Step	Candidates	Discarded	Confirmed
Candidate generation	800,447	0	0
Certify hyperbolicity	439,933	360,514	0
Regina census lookup	11,540	24	428,369
Essential S^2 or torus	11	0	11,529
Identified to be m135(1,3)	0	11	0
Total	800,447	360,549	439,898

Applying `regina.Triangulation3.isSphere` on fillings in Theorem 14 reveals that

▶ **Theorem 15.** *There are precisely* 1,849 *manifolds in the* 10*-tetrahedra census that are exteriors of knots in* S^3*.*

For censuses of at most 9 tetrahedra, similar work have been done by Callahan, Dean and Weeks [10], Champanerkar, Kofman and Patterson [12] and Dunfield [17].

Many conjectures and results mentioned in [17] can be tested and updated against the extended exceptional census. We plan to address these in a future publication.

4.2 Thurston norm and twisted Alexander polynomial

Given an orientable cusped hyperbolic 3-manifold M with Betti number $b_1(M) = 1$ and an $\mathrm{SL}(2,\mathbb{C})$-lift $\tilde{\rho}$ of its holonomy representation $\rho\colon \pi_1(M) \to \mathrm{PSL}(2,\mathbb{C})$, we consider its twisted Alexander polynomial $\tau_{2,\tilde{\rho}}(M) \in \mathbb{C}[t^{\pm 1}]$, defined by Dunfield, Friedl and Jackson [18, §2]. It was proved in [18] that

$$x(M) \geq \frac{1}{2} \deg \tau_{2,\tilde{\rho}}(M), \tag{2}$$

where $x(M)$ is the Thurston norm of a generator of $H^1(M;\mathbb{Z})$, and Equation (2) is an equality if M fibers over S^1. Moreover, if M fibers over S^1, then $\tau_{2,\tilde{\rho}}(M)$ is monic.

In [19], Dunfield, Garoufalidis and Yoon asked whether for any orientable cusped hyperbolic 3-manifold with $b_1(M) = 1$, there is at least one lift $\tilde{\rho}$ such that Equation (2) is an equality. An affirmative answer was then experimentally given for all such manifolds in the censuses of at most 9 tetrahedra, a sample size of 59,068 manifolds.

We extended the experimentation to the 10-tetrahedra census. There are 143,919 manifolds in the 10-tetrahedra census with $b_1(M) = 1$. By applying the code in [14], we were able to compute the Thurston norms for all except the following 21 of them:

$$\begin{array}{lllllll} o10_077959 & o10_076513 & o10_091925 & o10_073669 & o10_067720 & o10_102779 & o10_142965 \\ o10_141638 & o10_146022 & o10_140310 & o10_091511 & o10_122772 & o10_147820 & o10_127911 \\ o10_120303 & o10_145417 & o10_136855 & o10_136895 & o10_148034 & o10_143501 & o10_138268 \end{array} \tag{3}$$

Out of the 21 manifolds in Equation (3), the following 9 were confirmed to not fiber over S^1, as their twisted Alexander polynomials are not monic:

$$\begin{array}{lllll} o10_076513 & o10_091925 & o10_142965 & o10_091511 & o10_136855 \\ o10_136895 & o10_148034 & o10_143501 & o10_138268 & \end{array} \tag{4}$$

For the remaining 143,898 manifolds whose Thurston norms were successfully computed, we numerically computed their twisted Alexander polynomials of all $\mathrm{SL}(2,\mathbb{C})$-lifts. After taking the maximum degrees, we found that Equation (2) is an equality for all of them.

4.3 Homology 3-spheres and the L-space conjecture

The L-space conjecture asserts that for an irreducible $\mathbb{Q}$-homology 3-sphere Y, the following are equivalent:

(a) $\pi_1(Y)$ is left-orderable;
(b) Y has non-minimal Heegaard Floer homology;
(c) Y admits a taut foliation.

The equivalence of (a) and (b) was proposed by Boyer, Gordon and Watson [3]. The equivalence of (b) and (c) was asked by Ozsváth and Szabó [38] after they proved that (c) implies (b), and upgraded to a conjecture by A. Juhász [30]. For $\mathbb{Z}$-homology spheres, Calegari and Dunfield [9] proved that (c) implies (a).

Dunfield [16] showed that the L-space conjecture holds on at least 62.2% of 307,301 hyperbolic $\mathbb{Q}$-homology 3-spheres, obtained by Dehn fillings on orientable 1-cusped hyperbolic 3-manifolds in censuses of at most 9 tetrahedra. It was conjectured in [16] that these are precisely the filled manifolds from those censuses with lengths of systoles being at least 0.2.

While there are infinitely many hyperbolic $\mathbb{Q}$-homology 3-spheres obtained by Dehn fillings on a finite set of orientable 1-cusped hyperbolic 3-manifolds, a finite list is obtained if we bound the list by the length of systoles, i.e. the shortest geodesics, from below, as the following result was shown by Hodgson and Kerckhoff [26]:

▶ **Lemma 16** (Hodgson & Kerckhoff). *Let $\overline{M}$ be a compact orientable 3-manifold with torus boundary whose interior M is hyperbolic, and α be a slope on $\partial\overline{M}$. If $M(\alpha)$ is hyperbolic with systole length at least 0.163, then the normalized length of α is no more than 7.515.*

Here, the *normalized length* of α is the length of α divided by the area of $\partial(M \setminus N)$.

Similarly to Section 4.1, for the 1-cusped hyperbolic 3-manifolds in the 10-tetrahedra census, by enumerating all slopes with normalized length no more than 7.515, excluding all exceptional slopes, and filtering by the length of systole, we obtained a list of 1,899,974 fillings containing all pairs (M,α) such that $M(\alpha)$ is a hyperbolic $\mathbb{Q}$-homology 3-sphere with length of systole at least 0.163. Therefore,

▶ **Theorem 17.** *There are precisely 1,899,974 pairs (M,α) where M is a 1-cusped hyperbolic 3-manifold in the 10-tetrahedra census and α is a slope such that $M(\alpha)$ is a hyperbolic $\mathbb{Q}$-homology 3-sphere with length of systole at least 0.163.*

Changing $\mathbb{Q}$ to $\mathbb{Z}$ in Theorem 17, the number of pairs becomes 10,729.

Note that, although the pairs (M,α) in Theorem 17 are all distinct, the filled manifolds $M(\alpha)$ are not; for instance, there are at least 22 pairs in the list such that $M(\alpha)$ is identical to m007(3,1). It remains to group the pairs into distinct isometry classes by $M(\alpha)$, compare the grouped list against the list of 307,301 manifolds in [16], and test the L-space conjecture on it. We wish to address these in a future publication.

4.4 Closed totally geodesic surfaces

An embedded surface S in a hyperbolic 3-manifold M is called *totally geodesic* if every geodesic in S is also a geodesic in M. The existence of closed totally geodesic surfaces is of particular interest.

An algorithm that is guaranteed to find all closed totally geodesic surfaces (up to isotopy) in a given orientable cusped finite-volume hyperbolic 3-manifold was built and used to find several examples of cusped hyperbolic 3-manifolds with closed totally geodesic surfaces by Basilio, Lee and Malionek [2]; however, none was found in the censuses of at most 9 tetrahedra. By applying their algorithm to the 10-tetrahedra census, we found that

▶ **Theorem 18.** *The* 2*-cusped manifold o10_143602, whose volume is approximately* 9.1345 *and first homology group is* $\mathbb{Z}^2 \oplus \mathbb{Z}_6$*, is the only manifold containing closed totally geodesic surfaces in the orientable cusped hyperbolic censuses of at most* 10 *tetrahedra. Moreover, o10_143602 contains exactly one such surface up to isotopy, which is nonorientable with Euler characteristic* -1*.*

In particular, among all orientable cusped hyperbolic 3-manifolds containing closed totally geodesic surfaces, o10_143602 is the one whose minimal triangulations have the fewest tetrahedra; in other words, it is the simplest such example.

References

1. Ian Agol. Bounds on exceptional Dehn filling. *Geom. Topol.*, 4(1):431–449, 2000. doi:10.2140/gt.2000.4.431.
2. Brannon Basilio, Chaeryn Lee, and Joseph Malionek. Totally Geodesic Surfaces in Hyperbolic 3-Manifolds: Algorithms and Examples. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:19, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.14.
3. Steven Boyer, Cameron McA Gordon, and Liam Watson. On L-spaces and left-orderable fundamental groups. *Math. Ann.*, 356(4):1213–1245, 2013. doi:10.1007/s00208-012-0852-7.
4. Benjamin A. Burton. The Pachner graph and the simplification of 3-sphere triangulations. In *Computational geometry (SCG'11)*, pages 153–162. ACM, New York, 2011. doi:10.1145/1998196.1998220.
5. Benjamin A. Burton. Computational topology with Regina: algorithms, heuristics and implementations. In *Geometry and topology down under*, volume 597 of *Contemp. Math.*, pages 195–224. Amer. Math. Soc., Providence, RI, 2013. doi:10.1090/conm/597/11877.
6. Benjamin A. Burton. A duplicate pair in the SnapPea census. *Exp. Math.*, 23(2):170–173, 2014. doi:10.1080/10586458.2014.886535.
7. Benjamin A. Burton. The cusped hyperbolic census is complete. *Trans. Am. Math. Soc.*, pages 1–, 2017. doi:10.1090/tran/6767.
8. Benjamin A. Burton. The next 350 million knots. In *36th International Symposium on Computational Geometry*, volume 164 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 25, 17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.SoCG.2020.25.
9. Danny Calegari and Nathan M. Dunfield. Laminations and groups of homeomorphisms of the circle. *Invent. Math.*, 152(1):149–204, 2003. doi:10.1007/s00222-002-0271-6.
10. Patrick J. Callahan, John C. Dean, and Jeffrey R. Weeks. The simplest hyperbolic knots. *J. Knot Theory Ramifications*, 8(3):279–297, 1999. doi:10.1142/S0218216599000195.
11. Patrick J. Callahan, Martin V. Hildebrand, and Jeffrey R. Weeks. A census of cusped hyperbolic 3-manifolds. *Math. Comp.*, 68(225):321–332, 1999. With microfiche supplement. doi:10.1090/S0025-5718-99-01036-4.
12. Abhijit Champanerkar, Ilya Kofman, and Eric Patterson. The next simplest hyperbolic knots. *J. Knot Theory Ramifications*, 13(7):965–987, 2004. doi:10.1142/S021821650400355X.
13. Marc Culler, Nathan M. Dunfield, Matthias Goerner, and Jeffrey R. Weeks. SnapPy, a computer program for studying the geometry and topology of 3-manifolds. Available at http://snappy.computop.org (28/10/2025).

14 Nathan Dunfield, Stavros Garoufalidis, and Seokbeom Yoon. Code and data for comparing 1-loop invariants and torsions, 2024. doi:10.7910/DVN/XUJ3RH.

15 Nathan M. Dunfield. Cyclic surgery, degrees of maps of character curves, and volume rigidity for hyperbolic manifolds. *Invent. Math.*, 136(3):623–657, 1999. doi:10.1007/s002220050321.

16 Nathan M Dunfield. Floer homology, group orderability, and taut foliations of hyperbolic 3–manifolds. *Geom. Topol.*, 24(4):2075–2125, 2020. doi:10.2140/gt.2020.24.2075.

17 Nathan M. Dunfield. A census of exceptional Dehn fillings. In *Characters in low-dimensional topology*, volume 760 of *Contemp. Math.*, pages 143–155. Amer. Math. Soc., [Providence], RI, [2020] ©2020. doi:10.1090/conm/760/15289.

18 Nathan M. Dunfield, Stefan Friedl, and Nicholas Jackson. Twisted Alexander polynomials of hyperbolic knots. *Exp. Math.*, 21(4):329–352, 2012. doi:10.1080/10586458.2012.669268.

19 Nathan M Dunfield, Stavros Garoufalidis, and Seokbeom Yoon. 1-loop equals torsion for fibered 3-manifolds. *arXiv.org*, 2024. URL: https://arxiv.org/abs/2304.00469.

20 Nathan M. Dunfield, Neil R. Hoffman, and Joan E. Licata. Asymmetric hyperbolic L-spaces, Heegaard genus, and Dehn filling. *Math. Res. Lett.*, 22(6):1679–1698, 2015. doi:10.4310/MRL.2015.v22.n6.a7.

21 D. B. A. Epstein and R. C. Penner. Euclidean decompositions of noncompact hyperbolic manifolds. *J. Differential Geom.*, 27(1):67–80, 1988. doi:10.4310/jdg/1214441650.

22 Evgeny Fominykh, Stavros Garoufalidis, Matthias Goerner, Vladimir Tarkaev, and Andrei Vesnin. A census of tetrahedral hyperbolic manifolds. *Exp. Math.*, 25(4):466–481, 2016. doi:10.1080/10586458.2015.1114436.

23 Giles Gardam. Profinite rigidity in the SnapPea census. *Exp. Math.*, 30(4):489–498, 2021. doi:10.1080/10586458.2019.1577766.

24 Matthias Goerner. SnapPy: verified computations. Available at https://snappy.computop.org/verify.html (28/10/2025).

25 Martin Hildebrand and Jeffrey Weeks. A computer generated census of cusped hyperbolic 3-manifolds. In *Computers and mathematics (Cambridge, MA, 1989)*, pages 53–59. Springer, New York, 1989. doi:10.1007/978-1-4613-9647-5_7.

26 Craig D. Hodgson and Steven P. Kerckhoff. Universal bounds for hyperbolic Dehn surgery. *Ann. Math.*, 162(1):367–421, 2005. doi:10.4007/annals.2005.162.367.

27 Neil Hoffman, Kazuhiro Ichihara, Masahide Kashiwagi, Hidetoshi Masai, Shin'ichi Oishi, and Akitoshi Takayasu. Verified computations for hyperbolic 3-manifolds. *Exp. Math.*, 25(1):66–78, 2016. doi:10.1080/10586458.2015.1029599.

28 Jim Hoste, Morwen Thistlethwaite, and Jeff Weeks. The first 1,701,936 knots. *Math. Intelligencer*, 20(4):33–48, 1998. doi:10.1007/BF03025227.

29 William Jaco and Ulrich Oertel. An algorithm to decide if a 3-manifold is a Haken manifold. *Topology*, 23(2):195–209, 1984. doi:10.1016/0040-9383(84)90039-9.

30 András Juhász. A survey of Heegaard Floer homology. In *New ideas in low dimensional topology*, volume 56 of *Ser. Knots Everything*, pages 237–296. World Sci. Publ., Hackensack, NJ, 2015. doi:10.1142/9789814630627_0007.

31 Marc Lackenby. Word hyperbolic Dehn surgery. *Invent. Math.*, 140(2):243–282, 2000. doi:10.1007/s002220000047.

32 A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982. doi:10.1007/BF01457454.

33 Shana Yunsheng Li. The complete 10-tetrahedra census of orientable cusped hyperbolic 3-manifolds, 2025. arXiv:2512.02142.

34 Shana Yunsheng Li. The complete 10-tetrahedra census of orientable cusped hyperbolic 3-manifolds, 2025. doi:10.7910/DVN/BFEXZA.

35 Shana Yunsheng Li. snappy_11_tets, 2026. URL: https://pypi.org/project/snappy-11-tets/.

36 Feng Luo, Saul Schleimer, and Stephan Tillmann. Geodesic ideal triangulations exist virtually. *Proc. Amer. Math. Soc.*, 136(7):2625–2630, 2008. doi:10.1090/S0002-9939-08-09387-8.

37 Bruno Martelli. An introduction to geometric topology, 2022. arXiv:1610.02592.

38 Peter Ozsváth and Zoltán Szabó. Holomorphic disks and knot invariants. *Adv. Math.*, 186(1):58–116, 2004. doi:10.1016/j.aim.2003.05.001.

39 Stuart Rankin, Ortho Flint, and John Schermann. Enumerating the prime alternating knots, Part I. *J. Knot Theory Ramifications*, 13(1):57–100, 2004. doi:10.1142/S0218216504003044.

40 Makoto Sakuma and Jeffrey R. Weeks. The generalized tilt formula. *Geom. Dedicata*, 55(2):115–123, 1995. doi:10.1007/BF01264924.

41 Peter Guthrie Tait. Xix.—on knots. Part II. *Transactions of the Royal Society of Edinburgh*, 32(2):327–342, 1884. doi:10.1017/S0080456800026806.

42 Morwen Thistlethwaite. Index of /∼mthistle/8tet. Available at https://web.math.utk.edu/~mthistle/8tet/ (04/02/2026).

43 Morwen B. Thistlethwaite. The enumeration and classification of prime 20-crossing knots. *Algebr. Geom. Topol.*, 25(1):329–344, 2025. doi:10.2140/agt.2025.25.329.

44 Jeffrey R. Weeks. Convex hulls and isometries of cusped hyperbolic 3-manifolds. *Topology Appl.*, 52(2):127–149, 1993. doi:10.1016/0166-8641(93)90032-9.

Manifolds of Positive Reach, Differentiability, Tangent Variation, and Attaining the Reach

André Lieutier
Aix-en-Provence, France

Mathijs Wintraecken
Inria Centre d'Université Côte d'Azur, Sophia Antipolis, France

Abstract

This paper contains three main results.

Firstly, we give an elementary proof of the following statement: Let $\mathcal{M}$ be a topological manifold without boundary embedded in $\mathbb{R}^d$. If $\mathcal{M}$ has positive reach, then $\mathcal{M}$ can locally be written as the graph of a $C^{1,1}$ function from the tangent space to the normal space. Conversely if $\mathcal{M}$ can locally be written as the graph of a $C^{1,1}$ function from the tangent space to the normal space, then $\mathcal{M}$ has positive reach. The result was hinted at by Federer when he introduced the reach, and proved by Lytchak. Lytchak's proof relies heavily on CAT(k)-theory. The proof presented here uses only basic results on homology.

Secondly, we give optimal Lipschitz-constants for the derivative, in other words we give an optimal bound for the angle between tangent spaces in term of the distance between the points. We stress that Lytchak did not provide any bound, let alone an optimal one, making his proof, although interesting from a mathematical perspective, ineffectual in an algorithmic setting. To provide precise and optimal bounds on the angle between tangent spaces, we formally introduce the local reach for sets of positive reach, based on Aamari et al.'s discussion for C^2 manifolds. We prove that the local reach of a manifold is completely characterized by the variation of tangent spaces. This improves earlier results, that were either suboptimal or assumed that the manifold was C^2.

Thirdly, we show that the value of the reach is equals minimum of the local reach of the set and a global bottleneck for any set. This generalizes a result by Aamari et al. which explains how the reach is attained for C^2 manifolds.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Reach, Manifolds, Differentiability class, Lipschitz continuity, Tangent space

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.74

Related Version *Full Version*: `https://hal.inria.fr/hal-04816588`

Funding *Mathijs Wintraecken*: Supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 754411. The Austrian science fund (FWF) grant No. M-3073, the ANR grant StratMesh (ANR-24-CE48-1899), and the welcome package from IDEX of the Université Côte d'Azur (ANR-15-IDEX-01).

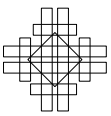
Acknowledgements We thank Jean-Daniel Boissonnat for discussion. We would also like to acknowledge the organizers of the workshop on "Algorithms for the Medial Axis", and Erin Chambers in particular for giving an impulse to this research. We further thank the reviewers for their comments that improved the exposition.

1 Introduction

In [13] Federer introduced the reach of a closed set $\mathcal{S} \subset \mathbb{R}^d$ as the minimum of the distance from $\mathcal{S}$ to the medial axis $\mathrm{ax}(\mathcal{S})$, i.e. the set of points in $\mathbb{R}^d$ for which the closest point in $\mathcal{S}$ is not unique. Assumptions on the reach (and its local version the local feature size [4]) underpin the correctness of many algorithms in computational geometry and topology. In



© André Lieutier and Mathijs Wintraecken;
licensed under Creative Commons License CC-BY 4.0



this paper we consider the differentiability class of manifolds of positive reach and give tight bounds on the angle between nearby tangent spaces for general manifolds of positive reach. For arbitrary sets of positive reach we give a geometrical explanation on how the reach in attained.

Previous work

Differentiability. Federer proved that the reach is stable under $C^{1,1}$-diffeomorphisms of the ambient space. Here we write $C^{1,1}$ to indicate C^1 maps whose derivative is Lipschitz, and by a $C^{1,1}$-diffeomorphism we mean that both the diffeomorphism and its inverse are $C^{1,1}$. Federer [13, Remark 4.20] furthermore mentioned (without going into much detail on one direction of the implication) that the graph of a function has positive reach if and only if it is $C^{1,1}$. Lytchak [21, 22] proved that a topologically embedded manifold without boundary has positive reach if and only if it is $C^{1,1}$, without a quantitative bound on the Lipschitz constant. This statement is quantified in our Theorem 1 below. We emphasize that quantifying these constants is essential for the result to allow us to formulate sampling criteria that guarantee the correctness of algorithms.

Lytchak's proof requires significant background in CAT(k)-theory. Lytchak, motivated by bounds on intrinsic curvature, does not give any quantified bound on the extrinsic curvature, which play an important role in this paper.

Scholtes [27] also gave a proof that a hypersurface has positive reach if and only if it is $C^{1,1}$, using different techniques from both the ones employed by Lytchak and in this paper. Scholtes' method uses the fact that the manifold has codimension one in an essential way.

Rataj and Zajíček [26] prove that if a Lipschitz manifold has positive reach, then it is $C^{1,1}$. They also use Lemma 17, however the assumption that the manifold is already Lipschitz simplifies the matter considerably, in particular they can skip the topological analysis. Moreover, again, they do not quantify their results in the way that we do and is essential for guarantees for algorithms. Recently, Leobacher and Steinicke [18] also reproved Lytchak's result, but once more the result was not quantified. We further stress that the bounds that we provide on the Lipschitz constants are tight, as we'll discuss in more detail below.

Geometric bounds and tangent variation. Bounding the tangent angle variation on smooth manifolds is crucial for establishing constants in the context manifolds reconstruction and meshing [4, 10, 12, 7]. If we now focus on more recent results, some authors consider angle variation bounds for a given local feature size bound [17]. Other authors, including the ones of the present paper, give angle variation bounds based on the reach [8]. Assuming the surface to be smooth, more specifically C^2 smooth, allows using standard differential geometry tools such as second fundamental forms, curvatures and Riemannian geometry, while, as hinted at by Federer's and Lytchak's results, the weaker positive reach assumption is sufficient for many properties to hold.

For example, in [8], the angle variation bound given for C^2 manifolds are clearly optimal (the bound is attained on spheres) while only suboptimal bounds are derived for $C^{1,1}$, or positive reach, manifolds.

Attaining the reach. In [1, Theorem 3.4], Aamari et al. make the following observation for C^2 manifolds: The reach of a submanifold of Euclidean space can be expressed as the minimum of a global quantity, realized at so called "bottlenecks", and a local quantity, the inverse of the maximal extrinsic curvature. These "bottlenecks" can be formally related to a

global reach and the local quantity with a local reach, which we'll do in this paper. We stress that the concepts of the local and global reach are mentioned once in the introduction of [1], no formal definition was given in that paper. Moreover, while this assumption is somewhat implicit in the proof of [1, Theorem 3.4], the analysis uses the context of Riemannian geometry, assuming the C^2 regularity of the manifold. We further emphasize that [1] focused uniquely on manifolds, which contrasts with the definition for global reach and local reach for any set of positive reach in this paper.

These results explain how the reach is attained, see Figure 1. This gives insight in which parts of a set of positive reach present the "difficult" parts for algorithms, in the examples we'll discuss now.

Motivation and related work

The assumption of positive reach is central to almost all triangulation algorithms, see for example [10, 12, 7], manifold learning, see for example [15, 14, 2, 28], and homotopy inference, see for example [24, 29]. Closing the gap between C^2 and positive reach manifolds significantly extends the applications domains. This extension is necessary to include for example the boundary of objects that have been designed by computer aided design software. These boundaries are generically tangent continuous and have bounded curvature (e.g. two planar faces connected through a cylindrical fillet surface), but don't have continuous curvature.

The main result of this paper makes precise what it means for a topologically embedded submanifold of Euclidean space to have positive reach: The embedding is necessarily $C^{1,1}$-smooth (Theorem 1) and we give a tight bound on the (generalized) extrinsic curvature of the submanifold (Theorem 4).

Generalized tangent and normal spaces. Federer showed that although sets of positive reach are more general than smooth manifolds, they still posses generalized tangent and normal spaces, which are convex cones instead of just linear spaces. Roughly speaking, the normal cone at a point p of a set with positive reach determines the topological "link" of point p in the set, that is the local topology.

To prove the Theorem 1 it is essential to show that the normal cone at any point of a positive reach, topologically embedded n-manifold is a $(d-n)$-dimensional vector space. In the proofs of [21, 22] the relation between the normal cone at a point p of a topologically embedded manifold with positive reach and the "link" of p is established using CAT(k)-theory. In contrast, our (short) proof (see the overview in Section 3 and Lemma 31 of [20]) is based on elementary properties of (the homology of) convex cones makes this correspondence more transparent in our opinion.

We further extend the results of [21, 22] by giving an optimal bound on the angle variation of tangent spaces. This also extends the same bound obtained in the particular context of C^2 manifolds [8]. We even obtain a significantly stronger result, that is, we characterize the local reach of a manifold completely by the tangent variation.

Contribution. In this paper we give an elementary proof of the following characterizations of manifolds of positive reach. Moreover the statement is quantified with optimal constants. Here we write $B(p,r)$ or $B^\circ(p,r)$ for the closed or open ball centred at p with radius r.

▶ **Theorem 1.** *If $\mathcal{M}$ is a topologically embedded manifold with positive reach and p a point in $\mathcal{M}$, then $\mathcal{M} \cap B^\circ(p, \sqrt{2}\operatorname{rch}(\mathcal{M})) \cap \pi^{-1}_{T_p\mathcal{M}}(B^\circ(0, \operatorname{rch}(\mathcal{M})))$ is the graph of a $C^{1,1}$ function above the open domain $B^\circ(0, \operatorname{rch}(\mathcal{M})) \subset T_p\mathcal{M}$.*

We also generalize the notations of local and global reach from C^2 submanifolds [1] to arbitrary compact sets (of positive reach):

▶ **Definition 2** (Local and global reach). *For a closed set $\mathcal{S} \subset \mathbb{R}^d$, we define the local* $\mathrm{rch}_{loc.}(\mathcal{S})$ *and global* $\mathrm{rch}_{glob.}(\mathcal{S})$ *reach as:*

$$\mathrm{rch}_{glob.}(\mathcal{S}) \underset{def.}{=} \inf\left\{ \frac{|a-b|}{2} \,\middle|\, \exists\, a,b \in \mathcal{S}, B^\circ\left(\frac{a+b}{2}, \frac{|a-b|}{2}\right) \cap \mathcal{S} = \emptyset \right\} \tag{1}$$

$$\mathrm{rch}_{loc.}(p,\mathcal{S}) \underset{def.}{=} \lim_{\substack{\rho \to 0 \\ \rho > 0}} \mathrm{rch}\big(\mathcal{S} \cap B(p,\rho)\big) \in [0,+\infty] \tag{2}$$

$$\mathrm{rch}_{loc.}(\mathcal{S}) \underset{def.}{=} \inf_{p \in \mathcal{S}} \mathrm{rch}_{loc.}(p,\mathcal{S}) \in [0,+\infty], \tag{3}$$

where $\inf(\emptyset) = +\infty$.

▶ Remark 3. If $\mathrm{rch}(\mathcal{S}) > 0$, then $\rho \mapsto \mathrm{rch}\big(\mathcal{S} \cap B(p,\rho)\big)$ is non-increasing (this follows from Theorem 16 because $\mathcal{S} \cap B(p,\rho)$ is geodesically convex [8, Corollary 1] for all $\rho < \mathrm{rch}(\mathcal{S})$ so that the set of bounds that need to be satisfied in (9) is larger, c.f. [6, Lemma 5]) as soon as $\rho < \mathrm{rch}(\mathcal{S})$, so that the limit in (2) exists. Moreover, since for any p one has $\mathrm{rch}_{loc.}(p,\mathcal{S}) \geq \mathrm{rch}(\mathcal{S})$, we get that:

$$\mathrm{rch}_{loc.}(\mathcal{S}) \geq \mathrm{rch}(\mathcal{S}). \tag{4}$$

To be able to give the bounds on the tangent variation/extrinsic curvature we need to introduce some notation.

We write $\mathrm{Gr}(n,\mathbb{R}^d)$ for the Grassmannian, that is the space of n-linear subspaces of $\mathbb{R}^d$ with metric given by the sine of the maximum angle. We recall that the maximum angle is given by

$$\angle A, B \underset{\text{def.}}{=} \max_{a \in A \setminus \{0\}} \min_{b \in B \setminus \{0\}} \angle a, b = \max_{b \in B \setminus \{0\}} \min_{a \in A \setminus \{0\}} \angle a, b. \tag{5}$$

We use this maximal angle as our metric. It is not that difficult to establish that this is indeed a metric, see [30] or [25] for a far reaching generalization.[1]

▶ **Theorem 4.** *If $\mathcal{M} \subset \mathbb{R}^d$ is a topologically embedded n-manifold with reach at least $R > 0$, then tangent spaces and normal spaces to $\mathcal{M}$ are linear spaces at any point $p \in \mathcal{M}$ and the maps* $\mathrm{Tan} : \mathcal{M} \to \mathrm{Gr}(n,\mathbb{R}^d)$ *and* $\mathrm{Nor} : \mathcal{M} \to \mathrm{Gr}(d-n,\mathbb{R}^d)$ *are Lipschitz.*

Moreover, the smallest Lipschitz constant satisfied by $p \mapsto \mathrm{Tan}(p,\mathcal{S})$ is $\frac{1}{\mathrm{rch}_{loc.}(\mathcal{M})}$:

$$\sup_{\substack{p,q \in \mathcal{M} \\ p \neq q}} \frac{\angle\, \mathrm{Tan}(p,\mathcal{M}), \mathrm{Tan}(q,\mathcal{M})}{d_{\mathcal{M}}(p,q)} = \frac{1}{\mathrm{rch}_{loc.}(\mathcal{M})}, \tag{6}$$

where $d_{\mathcal{M}}$ denotes the geodesic distance on $\mathcal{M}$ and we say that $d_{\mathcal{M}}(p,q) = \infty$ when p and q are not connected by any path in $\mathcal{M}$ and that $1/\infty = 0$.

[1] The sine of this angle also defines a metric on the Grassmannian as can be found in e.g. [16, Chapter IV, Section 2.1] or [3, Section 34], see Appendix B of [20] for a more extensive description. It seems that in a number of fields the sine of the angles is a more common metric, but it is suboptimal in this context. We further stress that neither of these metrics equal the standard Riemannian metric on the Grassmannian, which is given by the sum of the squares of the principal angles, see e.g. [31].

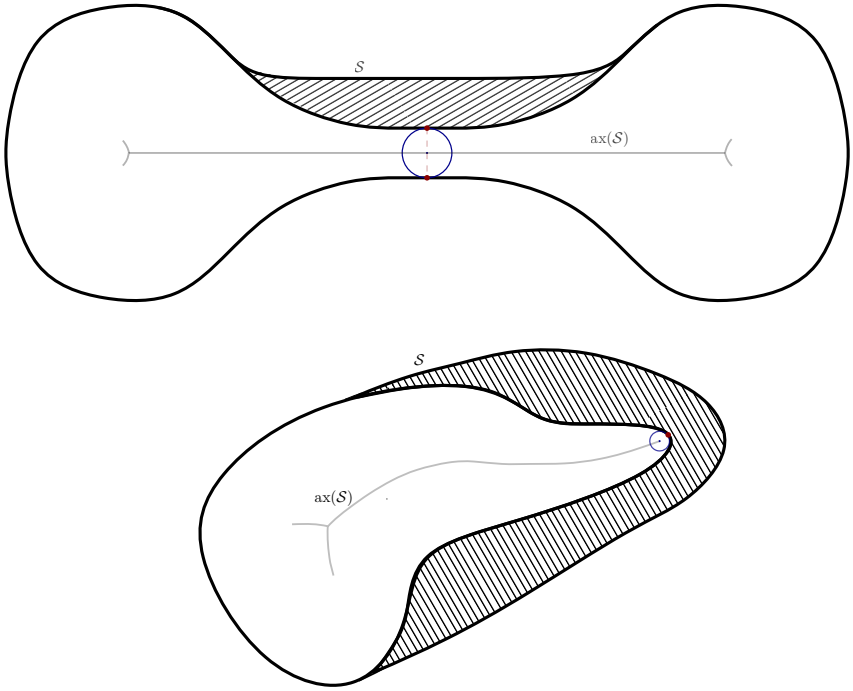


Figure 1 In this figure we see the two ways in which the reach can be attained for a general set of positive reach, by means of the global reach (top) or local reach (bottom), for two sets of positive reach in the plane. The shading indicates a solid set. The points (red) and the circle (blue) in which the reach is attained are indicated.

▶ Remark 5. If $\mathcal{M} \subset \mathbb{R}^d$ is a manifold with boundary then the results of Theorem 4 still hold as long as there is a shortest path geodesic between p and q that lies completely in the interior of $\mathcal{M}$.

Next theorem is a generalization of [1, Theorem 3.4] which was formulated in the particular situation where $\mathcal{S}$ is a C^2 embedded manifold, in which case $\mathrm{rch}_{loc.}(\mathcal{S})$ is also the supremum of extrinsic curvature.

▶ **Theorem 6.** *If $\mathcal{S}$ is a compact subset of $\mathbb{R}^d$, then:*

$$\mathrm{rch}(\mathcal{S}) = \min(\mathrm{rch}_{glob.}(\mathcal{S}), \mathrm{rch}_{loc.}(\mathcal{S}))$$

We refer to Figure 1 for an illustration.

2 Preliminaries

In this section we first recall some results and definitions concerning sets of positive reach. The results concerning the reach are mainly taken from [13] and [8].

2.1 Sets of positive reach

Throughout this paper we will write $\mathcal{S}$ for a general closed set (mostly with positive reach) and $\mathcal{M}$ for a manifold (mostly without boundary) and in some (rare) cases $\partial\mathcal{M}$ for its boundary (if it exists).

The concept of a tangent space of a smoothly embedded manifold has been generalized by Federer to tangent cones in the following manner:

▶ **Definition 7** (Generalized tangent spaces, Definitions 4.3 and 4.4 of [13]). *If $\mathcal{S} \subset \mathbb{R}^d$ and $p \in \mathcal{S}$, then the generalized tangent space*

$$\mathrm{Tan}(p, \mathcal{S})$$

is the set of all tangent vectors of $\mathcal{S}$ at p consists of all those $u \in \mathbb{R}^d$, such that either $u = 0$ or for every $\epsilon > 0$ there exists a point $q \in \mathcal{S}$ with

$$0 < |q - p| < \epsilon \qquad \text{and} \qquad \left| \frac{q-p}{|q-p|} - \frac{u}{|u|} \right| < \epsilon.$$

The set

$$\mathrm{Nor}(p, \mathcal{S})$$

of all normal vectors of $\mathcal{S}$ at p consists of all those $v \in \mathbb{R}^d$ such that $v \cdot u \leq 0$ for all $u \in \mathrm{Tan}(p, \mathcal{S})$.

▶ **Definition 8.** *We write $\pi_{\mathcal{S}}$ for the closest point projection on $\mathcal{S}$.*

Although it is possible to define a set valued version of $\pi_{\mathcal{S}}$, we only use the projection map outside the medial axis, that is at those points where there is a single closest point.

For the remainder of this subsection we will recall the properties of this map and sets of positive reach that are essential for this paper.

▶ **Lemma 9** (Angle with tangent space, Theorem 4.8(7) of [13]). *Suppose that $\mathcal{S}$ is a set positive reach $\mathrm{rch}(\mathcal{S})$. Let $p, q \in \mathcal{S} \subset \mathbb{R}^d$ such that $|p - q| < \mathrm{rch}(\mathcal{S})$. We have*

$$\sin \angle(q - p, \mathrm{Tan}(p, \mathcal{S})) \leq \frac{|p - q|}{2\,\mathrm{rch}(\mathcal{S})}. \tag{7}$$

Moreover, a variant of the previous lemma characterizes the reach for arbitrary closed sets, that is:

▶ **Lemma 10** (Distance to tangent space characterizes the reach, Theorem 4.18 of [13]). *For a closed subset $\mathcal{S}$ of Euclidean space $\mathbb{E} = \mathbb{R}^d$, one has:*

$$\mathrm{rch}(\mathcal{S}) = \inf_{\substack{p,q \in \mathcal{S} \\ p \neq q}} \frac{|p - q|^2}{2 d_{\mathbb{E}}(q,\ p\ + \mathrm{Tan}(p, \mathcal{M}))} \tag{8}$$

▶ **Lemma 11** (Theorem 4.8 (4) of [13]). *$\pi_{\mathcal{S}}$ is continuous on $\mathbb{R}^d \setminus \mathrm{ax}(\mathcal{S})$.*

In fact the following lemma strengthens the previous.

▶ **Lemma 12** (Theorem 4.8 (8) of [13]). *If $x, y \in \mathbb{R}^d$, and*

$$\max\{|x - \pi_{\mathcal{S}}(x)|, |y - \pi_{\mathcal{S}}(y)|\} = \mu \leq \mathrm{rch}(\mathcal{S}),$$

then

$$|\pi_{\mathcal{S}}(x) - \pi_{\mathcal{S}}(y)| \leq \frac{\mathrm{rch}(\mathcal{S})}{\mathrm{rch}(\mathcal{S}) - \mu} |x - y|$$

▶ **Lemma 13** (Theorem 4.8 (12) of [13]). *If $p \in \mathcal{S}$ and $lfs(p) > \rho > 0$, then*

$$\mathrm{Nor}(p, \mathcal{S}) = \{\lambda v \mid \lambda \geq 0, |v| = \rho, \pi_{\mathcal{S}}(p + v) = p\}.$$

$\mathrm{Tan}(p, \mathcal{S})$ *is the convex cone dual to* $\mathrm{Nor}(p, \mathcal{S})$*, and*

$$\lim_{t \to 0^+} t^{-1} d(p + tu, \mathcal{M}) = 0,$$

for $u \in \mathrm{Tan}(p, \mathcal{S})$, where d is the Euclidean distance.

▶ **Lemma 14** (Remark 4.15 (1) of [13]). *If $p \in \mathcal{S}$ and $lfs(p) > \rho > 0$, then $\mathcal{S} \cap B(p, \rho)$ is contractible.*

2.1.1 Geodesics

We also recall a result from the second paragraph of Part III, Section 1: "Die Existenz geodätischer Bogen in metrischen Räumen" in [23]:

▶ **Lemma 15** (Menger's existence of geodesics). *For a closed set $\mathcal{S} \subset \mathbb{R}^d$, $d_{\mathcal{S}}$ denotes the geodesic distance in $\mathcal{S}$, i.e. $d_{\mathcal{S}}(a, b)$ is the infimum of lengths of paths in $\mathcal{S}$ between a and b. If there is at least one path between a and b with finite length, then it is known that a minimizing geodesic, i.e. a path with minimal length connecting a to b exists. When no such path between a and b with finite length exists, we write $d_{\mathcal{S}}(a, b) = \infty$.*

We'll be using the metric characterization of the reach from [8] quite often in this paper.

▶ **Theorem 16** (Theorem 1 of [8]). *If $\mathcal{S} \subset \mathbb{R}^d$ is a closed set, then*

$$\operatorname{rch} \mathcal{S} = \sup \left\{ r > 0,\ \forall a, b \in \mathcal{S},\ |a - b| < 2r \Rightarrow d_{\mathcal{S}}(a, b) \leq 2r \arcsin \frac{|a - b|}{2r} \right\}, \tag{9}$$

where the sup *over the empty set is* 0*.*

2.2 Embeddings and atlases

We consider below an embedded, compact n-manifold $\mathcal{M} \subset \mathbb{R}^d$ with an atlas associated with an open cover $(U_i)_{i \in I}$:

$$\mathcal{M} = \bigcup_{i \in I} U_i$$

and charts $f_i : U_i \to \mathbb{R}^n$ that are homeomorphisms on their images, where the topology on $\mathcal{M}$ is the one induced by the ambient metric of $\mathbb{R}^d$.

Note that, topologically[2] embedded manifold does not exclude wild embeddings [11] but, as seen below, the positive reach assumption does, because the embedding is proven to be $C^{1,1}$.

[2] We say that an embedding is topological to stress that we don't make any assumption beyond the assumption that it is a homeomorphism onto its image, in particular no assumptions on differentiability are made.

3 Overview of the first two main results: Manifolds of positive reach, differentiability, and tangent variation

In this section we provide an overview of the proof of the first two main results of this paper Theorems 1 and 4. We discuss this in complete detail with full proofs of the statements in the appendix of the full version of this paper, that is [20].

A topologically embedded manifold with positive reach is differentiably embedded. Consider a topologically embedded manifold with positive reach. That is, we don't assume anything on the embedding of the manifold except that it is a homeomorphism with its image. However, the assumption that the manifold has positive reach yields that the generalized tangent spaces (see Definition 7) are cones.

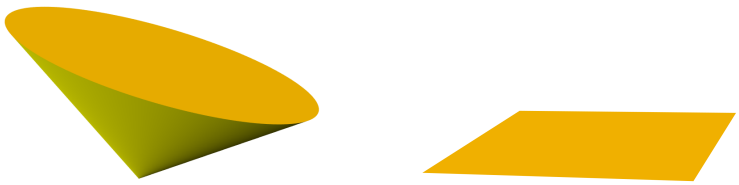


Figure 2 Convex cones come in two variants, a tapering object or a half space on the one hand and vector spaces on the other.

We characterize convex cones by topological means (roughly speaking) as follows. We recall that a convex cone is a subset of $\mathbb{R}^d$ that is convex and for each v in the set we have that λv is also in the set for $\lambda \in \mathbb{R}_{\geq 0}$. By this we mean that, roughly speaking, convex cones A come into two classes:

1. The convex cone is defined by

   $$A = \{\lambda a | \lambda \geq 0, a \in \tilde{A}\},$$

   where $\tilde{A}$ is a contractible subset of $\mathbb{S}^{d-1}$. Note that $\tilde{A} = A \cap \mathbb{S}^{d-1}$. In this case A is (again roughly speaking) either a tapering object or a half space.
2. The convex cone is defined by

   $$A = \{\lambda a | \lambda \geq 0, a \in \tilde{A}\},$$

   where $\tilde{A}$ is a (k dimensional, with $0 \leq k \leq d-1$) geodesic sphere in $\mathbb{S}^{d-1}$. In this case A is a vector space.

These two classes are characterized by the homology of $B(0,1) \setminus A^{\perp}$. The contractibility of $\tilde{A}$ is the key ingredient, see Lemma 30 of [20]. More precisely, we'll use that $\tilde{A}$ is contractible on the sphere if and only if A isn't a vector space.

In Section A.2 of the appendix of [20] we'll establish, based on the characterization of Appendix A.1 of [20], that for topologically embedded manifolds the tangent cones are in fact tangent spaces. The proof proceeds in two steps:

- In Lemma 31 of [20], we'll see that $B(p,\rho) \setminus \big(p + \mathrm{Nor}(p,\mathcal{S})\big)$ has the same homology as $\mathcal{S} \cap B(p,\rho) \setminus \{p\}$, see Figure 3. This result will rely on the deformation retract along the closest point projection of sets of positive reach and its continuity as established by Federer [13]. We stress that the normal cone $\mathrm{Nor}(p,\mathcal{S})$ is the dual of the tangent cone $\mathrm{Tan}(p,\mathcal{S})$, which explains the link with Lemma 30 of [20].

- In the proof of Lemma 32 of [20], we'll see that if $\mathcal{M}$ is manifold, then $\mathcal{M} \cap B(p,\rho) \setminus \{p\}$ has the homology of a sphere of dimension $n-1$, which with the previous lemma allows us to conclude that $\mathrm{Tan}(p,\mathcal{M})$ is a vector space. The proof combines topological reasoning with arguments based on the charts of the manifold $\mathcal{M}$.

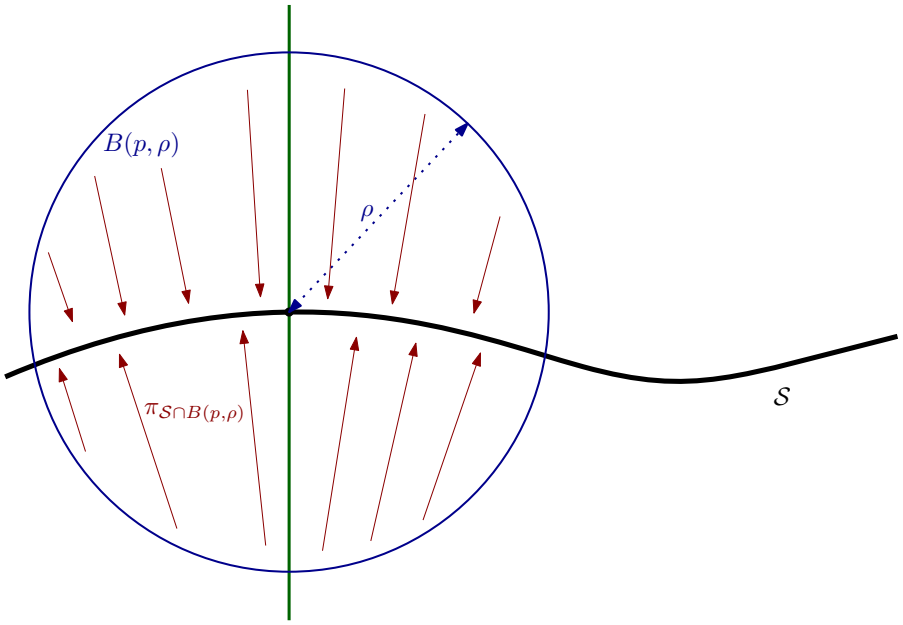


Figure 3 Sketch of the proof of Lemma 31 of [20]. We establish a homotopy between $B(p,\rho) \setminus \big(p + \mathrm{Nor}(p,\mathcal{S})\big)$ and $\mathcal{S} \cap B(p,\rho) \setminus \{p\}$ by deformation retract induced by the map $\pi_{\mathcal{S}\cap B(p,\rho)}$.

By definition we have that if a manifold has a tangent space it is differentiably embedded, but that does not mean that the manifold has to be continuously differentiably embedded, this is what will be the topic we'll address next.

Semi-continuity of normal cones and C^1 embeddings. In Section A.3 of [20] we will first prove a more general statement (Lemma 33 of [20]), namely semi-continuity of the normal cones for general sets of positive reach. By this we mean that given a point p and its normal cone $\mathrm{Nor}(p,\mathcal{S})$, for all ϵ there exists a neighbourhood of p such that for all p' in this neighbourhood, we have that the normal cone $\mathrm{Nor}(p',\mathcal{S})$ makes only an ϵ angle with some subset of $\mathrm{Nor}(p,\mathcal{S})$. The proof is elementary, but rather technical, and relies on compactness arguments and the manipulation of offsets.

Lemma 33 of [20] together with the observation that the tangent spaces are k-dimensional vector spaces the one sided bound on the angle between the normal cones above is in fact a two-sided bound, see Lemma 34 of [20], yields that the manifold is C^1, that is continuously differentiable. Lemma 35 of [20] strengthens this even further and explains that $\mathcal{M}$ can locally be written as the graph of a function. To put it differently, using some geometrical and analytical arguments we see that the manifold (locally) projects injectively on its tangent cone (which is an affine space by Lemma 32 of [20]). This then allows us to identify it, locally, as the graph of a differentiable map from the tangent space to the normal space.

From C^1 to $C^{1,1}$ embeddings. Appendix A.4 of [20] proves the majority of the main statements of the paper, namely that manifolds with positive reach are $C^{1,1}$ and characterizes the Lipschitz constants of the derivative in terms of the local reach. It is the most technical section and relies heavily on the theory of semiconcave functions. Most crucially we'll use the following result, namely Corollary 3.3.8 of [9]:

▶ **Lemma 17.** *Let $F : U \to \mathbb{R}$ be a function defined on a convex open set $U \subset \mathbb{R}^n$. Then the following properties are equivalent:*

- *F is of class $C^{1,1}$ with 1-Lipschitz derivative.*
- *both $x \to 1/2\, x^2 - F(x)$ and $x \to 1/2\, x^2 + F(x)$ are convex functions.*

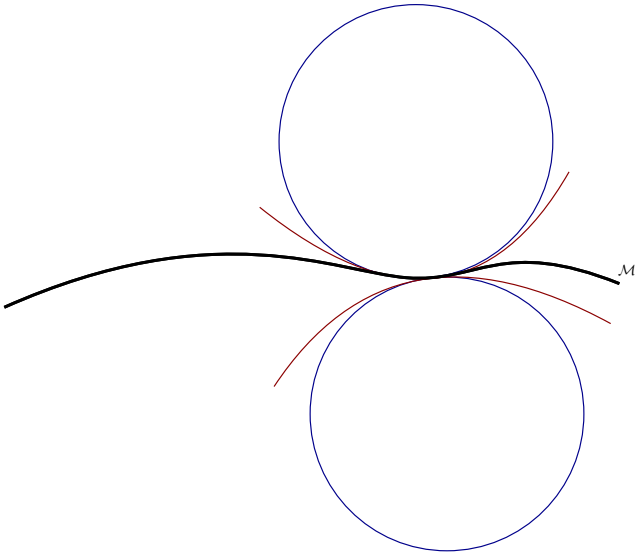


Figure 4 A manifold with empty tangent balls (blue) and (locally) empty tangent parabola (red).

Although that the precise technical way that this result is used is too intricate to explain in a paragraph or two, the link between sets of positive reach and Lemma 17 can be explained in geometrical terms. Assume for the moment that $\mathcal{M}$ has reach 1. As a consequence of Lemma 13 the interior of every ball that is tangent to a manifold $\mathcal{M}$, that has radius less or equal to 1 (the reach of $\mathcal{M}$) does not intersect $\mathcal{M}$ itself. Geometrically speaking the fact that $x \to 1/2\, x^2 - F(x)$ and $x \to 1/2\, x^2 + F(x)$ are convex functions can be expressed by the fact that (locally) the interior of parabolae that are tangent to the manifold have an empty interior, see Figure 4. Of course these parabolae are just second order approximations of the spheres. Because we are just interested in the Lipschitz constants of the derivative this approximation suffices.

Using Lemma 17 we do not directly prove our bound on the angle between nearby tangent spaces, instead we first establish the bound

$$\left\|D\phi(y_2) - D\phi(y_1)\right\| \leq \frac{1}{(\mathrm{rch}_{loc.}(p, \mathcal{M}) - \epsilon)} |y_2 - y_1|,$$

see Lemma 39 of [20], where ϕ is the local function (from the tangent to normal space) of which $\mathcal{M}$ is locally the graph and $\|\cdot\|$ denotes the operator two norm, in other words we regard the Jacobians as matrices depending on y and bound their Lipschitz constant in a way that is ϵ close to optimal.

From the result of Lemma 39 of [20] it is not very difficult to derive a local bound between tangent spaces, which is done in Lemma 40 of [20]. This local bound can then be extended (Lemma 41 of [20]) to a non-local statement by gluing the local estimates on the angle between tangent spaces along a geodesic, which exists thanks to Lemma 15. This allows us to establish one of the main results, the characterization of the local reach in terms of the tangent variation in Theorem 4, whose proof requires a significant amount of computation. The optimal size of the domain (in a fixed tangent space) such that $\mathcal{M}$ can be written as the graph of a function from a (fixed) tangent space to its normal space is then given in Theorem 1. The proof can be sketched as follows: By combining Theorems 4 and 16 with the submersion theorem we can establish a local diffeomorphism. The global diffeomorphism is established by proving that the covering number is one.

4 Global and local reach

In this section we stablish the third main result of the paper, that is Theorem 6. This theorem generalizes the results of [1] on how the reach is attained from C^2 manifolds to arbitrary sets of positive reach. It states that the reach is a consequence of either curvature maxima or so-called bottlenecks. In our result we replace the curvature by the local reach, which in turn is defined using local metric distortion, while the bottlenecks are similar in nature. In a number of cases we will have to assume that the set is bounded/compact.

Athough some of the details of the proof of Theorem 6 are a bit technical, the main idea is not that complicated. In fact we provide a sketch of the proof in Figure 5 and its caption.

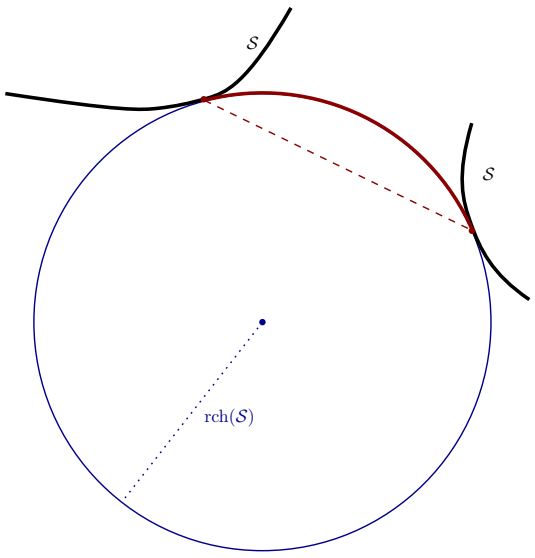


Figure 5 To derive a contradiction assume that $\mathrm{rch}(\mathcal{S}) < \mathrm{rch}_{loc.}(\mathcal{S})$ and $\mathcal{S}$ is tangent to the ball with radius $\mathrm{rch}(\mathcal{S})$ (blue) in two distinct points (red) that are closer than $2\mathrm{rch}(\mathcal{S})$, and therefore not antipodal. Consider the cord connecting these two points (dashed red), then Theorem 16 yields that the length of the geodesic is at most equal to part of the circle indicated in red, however that is the only curve of that length (or less) that connects the two red points that does not intersect the interior of the ball of radius $\mathrm{rch}(\mathcal{S})$. This implies that this curve has to lies inside $\mathcal{S}$, moreover this curve has local reach $\mathrm{rch}(\mathcal{S})$. Combining these observations yields that $\mathrm{rch}(\mathcal{S}) \geq \mathrm{rch}_{loc.}(\mathcal{S})$.

We recall the definition from the introduction:

▶ **Definition 2** (Local and global reach)**.** *For a closed set $\mathcal{S} \subset \mathbb{R}^d$, we define the local* $\mathrm{rch}_{loc.}(\mathcal{S})$ *and global* $\mathrm{rch}_{glob.}(\mathcal{S})$ *reach as:*

$$\mathrm{rch}_{glob.}(\mathcal{S}) \underset{def.}{=} \inf\left\{ \frac{|a-b|}{2} \,\middle|\, \exists\, a,b \in \mathcal{S}, B^\circ\left(\frac{a+b}{2}, \frac{|a-b|}{2}\right) \cap \mathcal{S} = \emptyset \right\} \tag{1}$$

$$\mathrm{rch}_{loc.}(p,\mathcal{S}) \underset{def.}{=} \lim_{\substack{\rho\to 0\\ \rho>0}} \mathrm{rch}\big(\mathcal{S}\cap B(p,\rho)\big) \in [0,+\infty] \tag{2}$$

$$\mathrm{rch}_{loc.}(\mathcal{S}) \underset{def.}{=} \inf_{p\in\mathcal{S}} \mathrm{rch}_{loc.}(p,\mathcal{S}) \in [0,+\infty], \tag{3}$$

where $\inf(\emptyset) = +\infty$.

▶ **Lemma 18.** *The map $p \mapsto \mathrm{rch}_{loc.}(p,\mathcal{S})$ defined over $\mathcal{S}$ is lower semi-continuous, in other words:*

$$\mathrm{rch}_{loc.}(p,\mathcal{S}) = \lim_{\substack{\rho\to 0\\ \rho>0}} \inf_{q\in\mathcal{S}\cap B^\circ(p,\rho)} \mathrm{rch}_{loc.}(q,\mathcal{S}).$$

Proof. By definition of $\mathrm{rch}_{loc.}(p,\mathcal{S})$, we have that for any $\epsilon > 0$ there is $\rho > 0$ such that:

$$\mathrm{rch}(p, \mathcal{S}\cap B(p,\rho)) > \mathrm{rch}_{loc.}(p,\mathcal{S}) - \epsilon. \tag{10}$$

One gets

$$\begin{aligned}
\mathrm{rch}_{loc.}(p,\mathcal{S}) &\geq \inf_{q\in\mathcal{S}\cap B^\circ(p,\rho)} \mathrm{rch}_{loc.}(q,\mathcal{S}) \\
&= \inf_{q\in\mathcal{S}\cap B^\circ(p,\rho)} \mathrm{rch}_{loc.}(q, \mathcal{S}\cap B(p,\rho)) \\
&\geq \mathrm{rch}_{loc.}(\mathcal{S}\cap B(p,\rho)) && \text{(by (3) applied to the set } \mathcal{S}\cap B(p,\rho)) \\
&\geq \mathrm{rch}(\mathcal{S}\cap B(p,\rho)) && \text{(by (4))} \\
&> \mathrm{rch}_{loc.}(p,\mathcal{S}) - \epsilon. && \text{(by (10))}
\end{aligned}$$

Because this inequality holds for any positive ϵ, the equality follows. ◀

▶ Remark 19. As a consequence, when $\mathcal{S}$ is compact, the inf in (3) is a min, in other words there exists at least one point $p \in \mathcal{S}$ such that $\mathrm{rch}_{loc.}(p,\mathcal{S}) = \mathrm{rch}_{loc.}(\mathcal{S})$.

▶ **Definition 20.** *A bottleneck is a pair of points $(a,b) \in \mathcal{S}^2$ such that*

$$B^\circ\left(\frac{a+b}{2}, \frac{|a-b|}{2}\right) \cap \mathcal{S} = \emptyset.$$

We have the following variant of Lemma 12 of [8], which in turn is a relatively straightforward generalization to arbitrary dimension of Property I of [5]:

▶ **Lemma 21.** *If a geodesic (or more generally a curve) γ parametrized according to arc length on the interval $[0,\ell]$ satisfies*

$$\angle\dot\gamma(s_1), \dot\gamma(s_2) \leq \frac{1}{R}|s_2 - s_1|, \tag{11}$$

then as long as $\ell \leq 2\pi R$,

$$|q-p| \geq 2\,\mathrm{rch}(\mathcal{M}) \sin\left(\frac{\ell}{2R}\right),$$

where $p = \gamma(0)$ and $q = \gamma(\ell)$.

Proof. The length of γ in the direction $\dot\gamma(\frac{\ell}{2})$ is

$$\begin{aligned}\left\langle q-p, \dot\gamma\left(\frac{\ell}{2}\right)\right\rangle &= \int_0^\ell \langle \dot\gamma(s), \dot\gamma(\ell/2)\rangle \mathrm{d}s \\ &= \int_0^{\ell/2} \langle \dot\gamma(s), \dot\gamma(\ell/2)\rangle \mathrm{d}s + \int_{\ell/2}^{\ell} \langle \dot\gamma(s), \dot\gamma(\ell/2)\rangle \mathrm{d}s \\ &\geq \int_0^{\ell/2} \cos\frac{|s-\ell/2|}{R}\mathrm{d}s + \int_{\ell/2}^{\ell} \cos\frac{|s-\ell/2|}{R}\mathrm{d}s \\ &= 2\,R \sin\left(\frac{\ell}{2R}\right).\end{aligned}$$

where, in the above inequality, we use the assumption $\ell \leq 2\pi R$ and the fact that $\theta \mapsto \cos\theta$ is decreasing on $[0,\pi]$.

Because $|q-p| \geq \langle q-p, \dot\gamma(\frac{\ell}{2})\rangle$, we see that

$$|q-p| \geq 2\,R \sin\left(\frac{\ell}{2R}\right). \qquad \blacktriangleleft$$

Combining Lemmas 42 of [20] and 21 we get:

▶ **Corollary 22.** *If γ is a geodesic in $\mathcal{S}$ with length $\ell \leq 2\pi \mathrm{rch}_{loc.}(\mathcal{S})$, then*

$$\ell \leq 2\mathrm{rch}_{loc.}(\mathcal{S}) \arcsin \frac{|p-q|}{2\mathrm{rch}_{loc.}(\mathcal{S})}.$$

The Lemma 23 below is a special case of [19, Lemma 4.6], we still provide a proof for completeness.

▶ **Lemma 23.** *Let $x \notin \mathrm{ax}(\mathcal{S})$ and denote by $\tilde x$ the unique point closest to x in $\mathcal{S}$. Then, for any $\epsilon > 0$, there is $\alpha > 0$ such that,*

$$|y-x| < \alpha \Rightarrow \{z \in \mathcal{S} \mid d(y,z) = d(y,\mathcal{S})\} \subset B^\circ(\tilde x, \epsilon). \tag{12}$$

Proof. Consider the closed set $\mathcal{S}' := \mathcal{S} \setminus B^\circ(\tilde x, \epsilon)$. The minimum distance $d(x,\mathcal{S}')$ to x is attained at least at some point z in $\mathcal{S}'$ so that

$$d(x,\mathcal{S}') = d(x,z) > d(x,\tilde x) = d(x,\mathcal{S}).$$

Taking

$$\alpha := \frac{d(x,\mathcal{S}') - d(x,\mathcal{S})}{2},$$

we get, for any y such that $|y-x| < \alpha$ and $z \in \mathcal{S}'$ that:

$$\begin{aligned} d(y,z) &\geq d(y,\mathcal{S}') \geq d(x,\mathcal{S}') - d(y,x) \\ &= d(x,\mathcal{S}) + 2\alpha - d(y,x) = d(x,\tilde x) + 2\alpha - d(y,x) \\ &> d(x,\tilde x) + d(y,x) \geq d(y,\tilde x) \geq d(y,\mathcal{S}). \end{aligned}$$

We have shown that if $z \in \mathcal{S}'$, then $d(y,z) > d(y,\mathcal{S})$. It follows that any closest point to y in $\mathcal{S}$ is in $\mathcal{S} \setminus \mathcal{S}' = \mathcal{S} \cap B^\circ(\tilde x, \epsilon)$ which is (12). ◀

▶ **Theorem 6.** *If $\mathcal{S}$ is a compact subset of $\mathbb{R}^d$, then:*

$$\mathrm{rch}(\mathcal{S}) = \min(\mathrm{rch}_{glob.}(\mathcal{S}), \mathrm{rch}_{loc.}(\mathcal{S}))$$

Proof. We consider first the situation where $\mathrm{rch}(\mathcal{S}) = 0$. In this case, since $\mathcal{S}$ is bounded, there exists some point $p \in \overline{\mathrm{ax}(\mathcal{S})} \cap \mathcal{S}$. Then, for any $\rho > 0$, there is some $p' \in \mathrm{ax}(\mathcal{S})$ such that $d(p', p) < \rho/2$, so that there are at least two points in $\mathcal{S} \cap B(p, \rho)$ closest to p'. Thus, $\mathrm{rch}(\mathcal{S} \cap B(p, \rho)) < \rho/2$, and, since this holds for any $\rho > 0$, we get that $\mathrm{rch}_{loc.}(p, \mathcal{S}) = 0$ and as a direct consequence $\mathrm{rch}_{loc.}(\mathcal{S}) = \inf_{q \in \mathcal{S}} \mathrm{rch}_{loc.}(q, \mathcal{S}) = 0$.

We assume now that $\mathrm{rch}(\mathcal{S}) > 0$.

As we have seen in Equation (4) of Remark 3, we have that $\mathrm{rch}(\mathcal{S}) \leq \mathrm{rch}_{loc.}(\mathcal{S})$. Therefore it suffices to prove that, if $\mathrm{rch}(\mathcal{S}) < \mathrm{rch}_{loc.}(\mathcal{S})$, then $\mathrm{rch}(\mathcal{S}) = \mathrm{rch}_{glob.}(\mathcal{S})$. So let us assume now that $\mathrm{rch}(\mathcal{S}) < \mathrm{rch}_{loc.}(\mathcal{S})$.

Since $\mathcal{S}$ is compact and thus bounded, which implies that a closed $\mathrm{rch}(\mathcal{S})$-neighbourhood of $\mathcal{S}$ is also bounded, there must exist $x \in \overline{\mathrm{ax}(\mathcal{S})}$, where $\overline{\mathrm{ax}(\mathcal{S})}$ is the closure of the medial axis $\mathrm{ax}(\mathcal{S})$, such that $d(x, \mathcal{S}) = \mathrm{rch}(\mathcal{S})$.

If $x \in \overline{\mathrm{ax}(\mathcal{S})} \setminus \mathrm{ax}(\mathcal{S})$, in particular $x \notin \mathrm{ax}(\mathcal{S})$ and there is a unique point $\tilde{x}$ closest to x in $\mathcal{S}$. Choose some $0 < \epsilon < \mathrm{rch}(\mathcal{S})$ and, as in Lemma 23, take $\alpha > 0$ so that (12) holds and, moreover, such that

$$\alpha < \mathrm{rch}_{loc.}(\mathcal{S}) - \mathrm{rch}(\mathcal{S}).$$

Since $x \in \overline{\mathrm{ax}(\mathcal{S})}$, there exists $y \in \mathrm{ax}(\mathcal{S}) \cap B^\circ(x, \alpha))$, and therefore Lemma 23 yields that there exists at least two points $p, q \in B^\circ(\tilde{x}, \epsilon)$ such that $d(y, p) = d(y, q) = d(y, \mathcal{S})$.

Because $d(p, q) < 2\epsilon < 2\mathrm{rch}(\mathcal{S})$, by Theorem 16, there is a geodesic from p to q of length at most $\pi\mathrm{rch}(\mathcal{S})$, which is less that $2\pi\mathrm{rch}_{loc.}(\mathcal{S})$. But then one can apply Corollary 22, which implies that the length of a geodesic from p to q is at most

$$\ell_{\max} = 2\mathrm{rch}_{loc.}(\mathcal{S}) \arcsin \frac{|p - q|}{2\mathrm{rch}_{loc.}(\mathcal{S})}.$$

However, since projecting a curve outside a given sphere onto the given sphere only decreases its length, a curve from p to q outside the ball centered at y whose radius is at most $\mathrm{rch}(\mathcal{S}) + |y - x| < \mathrm{rch}(\mathcal{S}) + \alpha < \mathrm{rch}_{loc.}(\mathcal{S})$ must have length strictly greater than $\ell_{\max}$, a contradiction.

We have so far proven that when $\mathrm{rch}(\mathcal{S}) < \mathrm{rch}_{loc.}(\mathcal{S})$, then the reach cannot be realized in $\overline{\mathrm{ax}(\mathcal{S})} \setminus \mathrm{ax}(\mathcal{S})$. One must then have $x \in \mathrm{ax}(\mathcal{S})$ and there are at least two points $p \neq q$ closest points to x on $\mathcal{S}$.

If $|p - q| < 2\mathrm{rch}(\mathcal{S})$ then by Theorem 16 one has that $d_{\mathcal{S}}(p, q) \leq 2\mathrm{rch}(\mathcal{S}) \arcsin \frac{|p-q|}{2\mathrm{rch}(\mathcal{S})}$ which is less than $2\pi\mathrm{rch}_{loc.}(\mathcal{S})$. Again, Corollary 22 implies that the length of a geodesic from p to q is a most $2\mathrm{rch}_{loc.}(\mathcal{S}) \arcsin \frac{|p-q|}{2\mathrm{rch}_{loc.}(\mathcal{S})}$, which, again, is impossible without entering the ball centered at x with radius $\mathrm{rch}(\mathcal{S}) < \mathrm{rch}_{loc.}(\mathcal{S})$.

Since $|p - q| \leq 2\mathrm{rch}(\mathcal{S})$, the only possibility is that $|p - q| = 2\mathrm{rch}(\mathcal{S})$, i.e. $x = (p + q)/2$, a bottleneck. ◀

References

1 Eddie Aamari, Jisu Kim, Frédéric Chazal, Bertrand Michel, Alessandro Rinaldo, and Larry Wasserman. Estimating the reach of a manifold. *Electronic journal of statistics*, 13(1):1359–1399, 2019.

2 Eddie Aamari and Clément Levrard. Stability and minimax optimality of tangential Delaunay complexes for manifold reconstruction. *Discrete & Computational Geometry*, 59(4):923–971, 2018. doi:10.1007/s00454-017-9962-z.

3 Naum Il'ich Akhiezer and Izrail Markovich Glazman. *Theory of linear operators in Hilbert space*. Courier Corporation, 2013.

4 N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, December 1999. doi:10.1007/PL00009475.

5 Dominique Attali, Herbert Edelsbrunner, and Yuriy Mileyko. Weak Witnesses for Delaunay triangulations of Submanifold. In *ACM Symposium on Solid and Physical Modeling*, pages 143–150, Beijing, China, June 2007. doi:10.1145/1236246.1236267.

6 Dominique Attali and André Lieutier. Geometry-driven collapses for converting a Čech complex into a triangulation of a nicely triangulable shape. *Discrete & Computational Geometry*, 54(4):798–825, 2015. doi:10.1007/S00454-015-9733-7.

7 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and Topological Inference*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2018. doi:10.1017/9781108297806.

8 Jean-Daniel Boissonnat, André Lieutier, and Mathijs Wintraecken. The reach, metric distortion, geodesic convexity and the variation of tangent spaces. *Journal of Applied and Computational Topology*, 3(1):29–58, June 2019. doi:10.1007/s41468-019-00029-8.

9 Piermarco Cannarsa and Carlo Sinestrari. *Semiconcave functions, Hamilton-Jacobi equations, and optimal control*, volume 58. Springer Science & Business Media, 2004.

10 S.-W. Cheng, T.K. Dey, and J.R. Shewchuk. *Delaunay Mesh Generation*. Computer and information science series. CRC Press, 2013.

11 Robert J Daverman and Gerard Venema. *Embeddings in manifolds*, volume 106. American Mathematical Soc., 2009.

12 T.K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Number 23 in Cambridge monographs on applied and computational mathematics. Cambridge University Press, 2007.

13 H. Federer. Curvature measures. *Transactions of the American mathematical Society*, 93:418–491, 1959.

14 Charles Fefferman, Sergei Ivanov, Yaroslav Kurylev, Matti Lassas, and Hariharan Narayanan. Reconstruction and interpolation of manifolds. I: The geometric Whitney problem. *Foundations of Computational Mathematics*, 2019. doi:10.1007/s10208-019-09439-7.

15 Charles Fefferman, Sergei Ivanov, Matti Lassas, and Hariharan Narayanan. Fitting a manifold of large reach to noisy data. *arXiv e-prints*, page arXiv:1910.05084, October 2019. arXiv:1910.05084.

16 Tosio Kato. *Perturbation theory for linear operators*, volume 132. Springer Science & Business Media, 2013.

17 Marc Khoury and Jonathan Richard Shewchuk. Approximation bounds for interpolation and normals on triangulated surfaces and manifolds. *arXiv preprint arXiv:1911.03424*, 2019. arXiv:1911.03424.

18 Gunther Leobacher and Alexander Steinicke. Existence, uniqueness and regularity of the projection onto differentiable manifolds. *Annals of Global Analysis and Geometry*, 60(3):559–587, July 2021. doi:10.1007/s10455-021-09788-z.

19 André Lieutier. Any open bounded subset of $\mathbb{R}^n$ has the same homotopy type as its medial axis. *Computer-Aided Design*, 36(11):1029–1046, 2004. Solid Modeling Theory and Applications. doi:10.1016/j.cad.2004.01.011.

20 André Lieutier and Mathijs Wintraecken. Manifolds of positive reach, differentiability, tangent variation, and attaining the reach. Full version, 2026. URL: https://inria.hal.science/hal-04816588.

21 Alexander Lytchak. On the geometry of subsets of positive reach. *manuscripta mathematica*, 115(2):199–205, 2004.

22 Alexander Lytchak. Almost convex subsets. *Geometriae Dedicata*, 115(1):201–218, 2005.

23 K. Menger. Untersuchungen über allgemeine Metrik. Vierte Untersuchung. Zur Metrik der Kurven. *Mathematische Annalen*, 103:466–501, 1930.

24 P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 39(1-3):419–441, 2008. doi:10.1007/S00454-008-9053-2.

25 Li Qiu, Yanxia Zhang, and Chi-Kwong Li. Unitarily invariant metrics on the grassmann space. *SIAM Journal on Matrix Analysis and Applications*, 27(2):507–531, 2005. doi:10.1137/040607605.

26 Jan Rataj and Luděk Zajíček. On the structure of sets with positive reach. *Mathematische Nachrichten*, 290(11-12):1806–1829, 2017.

27 S. Scholtes. On hypersurfaces of positive reach, alternating Steiner formulae and Hadwiger's Problem. *ArXiv e-prints*, April 2013. arXiv:1304.4179.

28 Barak Sober and David Levin. Manifold approximation by moving least-squares projection (MMLS). *Constructive Approximation*, pages 1–46, 2019. doi:10.1007/s00365-019-09489-8.

29 Yuan Wang and Bei Wang. Topological inference of manifolds with boundary. *Computational Geometry*, 88:101606, 2020. doi:10.1016/j.comgeo.2019.101606.

30 Per Åke Wedin. On angles between subspaces of a finite dimensional inner product space. In *Matrix Pencils: Proceedings of a Conference Held at Pite Havsbad, Sweden, March 22–24, 1982*, pages 263–285. Springer, 2006.

31 Yung-Chow Wong. Differential geometry of Grassmann manifolds. *Proceedings of the National Academy of Sciences of the United States of America*, 57(3):589, 1967.

Geodesics of Length Less Than πR in a Set of Reach R Are Unique and Continuous with Respect to the Endpoints

André Lieutier
Aix-en-Provence, France

Mathijs Wintraecken
Inria centre d'Université Côte d'Azur, Sophia Antipolis, France

Abstract

Positive reach underpins many results in computational geometry and topology. It is used for triangulation criteria, topological inference, and manifold learning. The geometric properties of these sets have therefore been studied intensely. Here we focus on the shortest paths or minimizing geodesics in these sets. Our main result states that minimizing geodesics of length strictly less than πR in a set of reach R are unique. This in turn implies that such minimizing geodesics are continuous with respect to the endpoints.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Reach, geodesics, metric geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.75

Funding *Mathijs Wintraecken*: Supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 754411. The Austrian science fund (FWF) M-3073. The welcome package from IDEX of the Université Côte d'Azur, ANR-15-IDEX-01. The French National Research Agency (ANR) under grant StratMesh, ANR-24-CE48-1899.

Acknowledgements We thank Jean-Daniel Boissonnat for discussion and Eddie Aamari for his insight in possible applications. We are indebted to the reviewers for many suggestion and in particular noting that Corollary 10 immediately follows from the results in our submission.

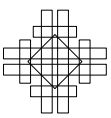
1 Introduction

Federer [14] introduced the reach of a closed set $\mathcal{S} \subset \mathbb{R}^d$ as the infimum of the distance from $\mathcal{S}$ to the medial axis $\mathrm{ax}(\mathcal{S})$, i.e. the set of points in $\mathbb{R}^d$ for which the closest point in $\mathcal{S}$ is not unique. Assumptions on the reach (and its local version the local feature size [3]) underpin the correctness of many algorithms in computational geometry and topology. In this paper we consider minimizing geodesics in this space and show that they are both unique and depend continuously on the endpoints if the geodesics are sufficiently short. Moreover, the bound on the length of the minimizing geodesics that implies uniqueness (and therefore continuity) is in fact tight, because it is attained on the sphere.

1.1 Motivation

Triangulation algorithms. The assumption of positive reach is central to almost all triangulation algorithms, see for example [10, 12, 5], as well as manifold learning, see for example [16, 15, 2, 30]. However these algorithms often consider manifolds of positive reach instead general sets of positive reach. If we want to extend our algorithm beyond the realm


© André Lieutier and Mathijs Wintraecken;
licensed under Creative Commons License CC-BY 4.0



of manifolds to more general stratified spaces, starting with "well behaved" sets of positive reach[1] or at least sets of which the strata have positive (local [24]) reach understanding the geometry of those sets seems essential.

Metric learning. Another motivation comes from metric learning. In [1] this is explained as follows:

"In the data analysis area, metric learning refers to the problem of finding a distance d over the space of observations that is relevant for a given task at stake [35, 32]. For instance, in a supervised framework where one is provided with tuples of allegedly similar or dissimilar observations, the goal is to find a distance that is small on the similar tuples and large on the dissimilar ones. There is a wide range of existing methods in the literature, ranging from parametric (LSI [34], MCML [18], LDML [21] among others) to nonparametric (DMLMJ [29], kernel methods [23, 9], to cite a few).

In an unsupervised setting, metric learning aims at finding a metric that takes into account the underlying geometry of the data. That is, it amounts to estimating the shortest-path (or geodesic) distance. Often, this is done via a dimension reduction technique: any low-dimensional embedding of the data gives rise to a new distance over the data in the embedded space. Existing algorithms include PCA, t-SNE [22], MDS [11], Isomap [33], or MVU [4]. See [32] for a thorough overview of the field."

And indeed in [1] geodesics play a crucial role in establishing optimal bounds.

1.2 Contribution

This paper focusses on the following result:

▶ **Theorem 1.** *Let $\mathcal{S}$ be a set of positive reach R, then for any two points $a, b \in \mathcal{S}$ such that the geodesic distance on $\mathcal{S}$ satisfies $d_{\mathcal{S}}(a, b) < \pi R$ there is a unique minimizing geodesic in $\mathcal{S}$ connecting a and b.*

We stress that in the context of sets of positive reach a minimizing geodesic is a curve of minimal length connecting the two points. In this paper we only consider minimizing geodesics and therefore will not explicitly repeat "minimizing" everywhere. More generally, a geodesic is a curve such that it is locally a minimizing geodesic (between the endpoints of a subsegment). In particular more than half of great circle on a sphere, is NOT a minimizing geodesic, although they are geodesics in the more general and Riemannian setting.

The following corollary to the main theorem is perhaps even more important for applications:

▶ **Corollary 2.** *Let $\mathcal{S}$ be a set of positive reach R, then the geodesic connecting any two points $a, b \in \mathcal{S}$ is continuous (in the C^0 sense) with respect to the endpoints provided that $d_{\mathcal{S}}(a, b) < \pi R$.*

Both results are tight, because the bound is attained on the sphere.

[1] Sets of positive reach are stratified spaces [14], where the stratification is understood in a weak sense. However, it is not clear that sets of positive reach are triangulable in general or additional assumptions are needed. This having been said, there are compact sets of (arbitrarily large) positive reach that are not (piecewise smoothly/linearly) homeomorphic to a finite simplicial complex, which puts this outside of the realm of computation.

1.3 Previous work

CAT(k) spaces. The terminology CAT(k) was introduced by Gromov [20] for metric spaces that satisfy some weak form of curvature bound. We'll now give a rough definition of CAT(k) spaces, but refer to [7] for a precise introduction, see Section II.1. Let $(\mathcal{M}, d_{\mathcal{M}})$ be a geodesic metric space, that is a space where the length between two points is defined as the minimum of all lengths of continuous (rectifiable) curves connecting the two points, i.e. the minimum is attained. Such a minimizing curve is called a geodesic. A geodesic triangle in such a space consists of three points and the geodesics connecting those points. We can then consider a geodesic triangle with the same edge lengths in a (simply connected) space of constant curvature k (that is a sphere, Euclidean space, or hyperbolic space). This triangle is called a comparison triangle. We say that the diameter of a simply connected space of constant curvature is $\pi/\sqrt{k}$ if $k > 0$ and ∞ otherwise. We also refer to this as the diameter of the comparison space. This geodesic triangle in $\mathcal{M}$ satisfies the CAT(k) inequality if the distances between the points on the triangle are less than the corresponding distances between points on the comparison triangle in a space of constant curvature k. If this holds for any geodesic triangle of diameter less than the diameter of the comparison space, then we say that the entire space is CAT(k).

Sets of positive reach as CAT(k) spaces. Lytchak [25, 26] proved sets of positive reach are CAT(k), for some k. Because in (simply connected) spaces of constant curvature, geodesic triangles with small diameter and one edge length equal to zero consist of two coinciding edges, the uniqueness of sufficiently small geodesics follows directly from Lytchak's result. However, from a computational perspective the result of Lytchak is ineffective, because we only know that such a constant exists and not what this constant may be.

Another way that this paper distinguishes itself compared to the literature, is that the results are all completely elementary, in the sense that we only need some results by Federer [14] and Euler-Lagrange theory [31].

1.4 Further motivation and future work

Many questions on the geometry of sets of positive reach and the behaviour of geodesics in sets of positive reach of course remain open. Determining the k for which a set of reach R is CAT(k), would be an important next step, which in fact motivated this work to a large extent. We conjecture that $k = 1/R^2$.

If we may engage in some speculation, establishing the CAT(k) property would potentially:

- Help establish bounds on angles between tangent cones of sets of positive reach, beyond the manifold setting, which is in turn essential for stratification learning.
- Help establish convergence rates for algorithms to compute geodesics in sets of positive reach.

2 Result

2.1 Notation

We'll denote sets of positive reach by $\mathcal{S}$ and the closest point projection by $\pi_{\mathcal{S}}$. We use the notation $B(c, r)$ for a ball in Euclidean spaces with centre c and radius r. Curves are generally denoted by γ and most importantly:

We'll use Newton's notation for the derivatives of curves that is, $\dot{\gamma} = \frac{d}{dt}\gamma$.

2.2 Uniqueness

The main idea of the proof is the following: Suppose that there are two geodesics γ_1 and γ_2 connecting two points is not unique then we can move the geodesic γ_1 a little in the direction γ_2, shortening the geodesic and thus yielding a contradiction. This perturbed geodesic, of course, needs to lie on the set of positive reach, which will be achieved by closest point projection onto the set of positive reach. We commence our proof with a couple of lemmas that will be instrumental in dealing with this projection.

We first recall a convexity result from [6]. We recall that a set C is geodesically convex if for any two points $c, c' \in C$ the geodesic connecting c and c is contained in C.

▶ **Corollary 3** (Corollary 1 of [6]). *Let $\mathcal{S} \subset \mathbb{R}^d$ be a closed set with positive reach $R > 0$. Then, for any $r' < R$ and any $x \in \mathbb{R}^d$ then $\mathcal{S} \cap B(x, r')$ is geodesically convex in $\mathcal{S}$.*

▶ **Lemma 4.** *Let $\mathcal{S}$ have reach R and assume $p, q \in \mathcal{S}$ with $|p - q| \leq 2R$. Define $x = \lambda p + (1 - \lambda)q$. We have that*

$$\begin{aligned} d(x, \mathcal{S}) &= |x - \pi_{\mathcal{S}}(x)| \\ &\leq R - \sqrt{R^2 - \lambda(1-\lambda)|p-q|^2} \\ &\leq \frac{\lambda(1-\lambda)|p-q|^2}{2R} + \mathcal{O}\left(\left(\frac{\lambda(1-\lambda)|p-q|^2}{2R}\right)^2\right), \end{aligned}$$

with d the Euclidean distance.

Proof. Corollary 3 implies that the geodesic connecting p and q is contained in the intersection of all balls of radius less than the reach that contain p and q so that in particular the geodesic is contained in the spindle (the intersection of all balls of radius R in Euclidean space that contain p and q), as depicted in Figure 1. This implies in turn that $|x - \pi_{\mathcal{S}}(x)|$ less than or equal to the distance from x to the boundary of the spindle. Let's restrict ourselves to the plane given by p, q, and $\pi_{\mathcal{S}}(x)$ and consider a circle of radius R with centre c that yields the boundary of the spindle in this plane, see again Figure 1. By the cosine rule have that

$$|c - \lambda p - (1-\lambda)q|^2 = R^2 + \lambda^2|p-q|^2 - 2R\lambda|p-q|\cos\theta,$$

where $\cos\theta = \frac{|p-q|}{2R}$ (the value for θ follows by inspection of the triangle with vertices p, c, and $\frac{1}{2}(p+q)$). So that

$$\begin{aligned} |c - \lambda p - (1-\lambda)q|^2 &= R^2 + \lambda^2|p-q|^2 - \lambda|p-q|^2 \\ &= R^2 - (1-\lambda)\lambda|p-q|^2. \end{aligned}$$

The first inequality now follows by observing that the distance from x to the spindle is given by $R - |c - \lambda p + (1-\lambda)q|$, see Figure 1. The second inequality follows from the standard observation that $\sqrt{1-t} = 1 - \frac{t}{2} - \mathcal{O}(t^2)$, which in turn follows by Taylor's theorem. ◀

We'll need the following corollary of [14, Theorem 4.8(8)]. Theorem 4.8(8) of [14] says the closest point projection $\pi_{\mathcal{S}}$ at a point x is $\frac{1}{1-d(x,\mathcal{S})/R}$-Lipschitz. We therefore immediately have:

▶ **Corollary 5.** *If $\gamma : [0, \ell] \to \mathbb{R}^d$ is a C^1 curve, then the length of $\pi_{\mathcal{S}}(\gamma)$ is upper bounded as follows*

$$\text{length}(\pi_{\mathcal{S}}(\gamma)) \leq \int_0^\ell \frac{1}{1 - d(\gamma(t), \mathcal{S})/R} |\dot{\gamma}(t)| dt. \tag{1}$$

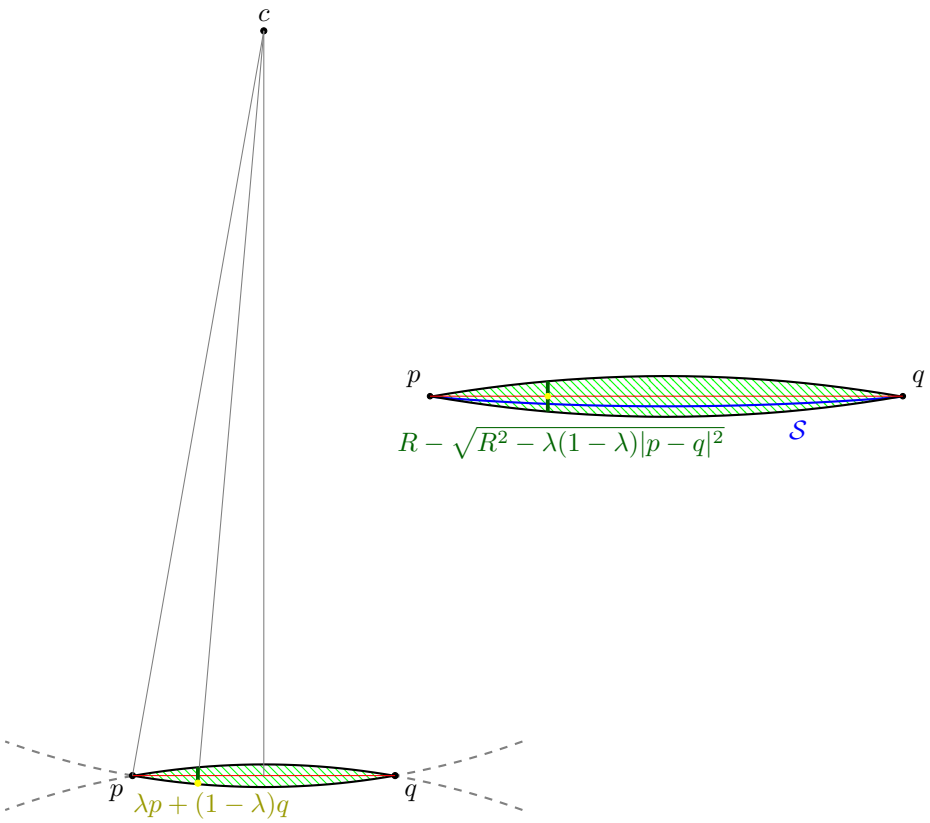


Figure 1 Left: The region in which the geodesic connecting p and q lies is indicated in green, the grey dashed circles have radius R. Right: A zoom in on the spindle with the distance from the yellow point $(\lambda p + (1-\lambda)q)$ to the boundary of the spindle indicated in dark green. The geodesic connecting p and q (which is a part of $\mathcal{S}$) within the spindle is indicated in blue. Because the union of the two dark green line segments (which are almost collinear in the figure, in higher dimensions it is a cone or in other words, the join of a sphere and a point), separates the spindle into two halves and the geodesic is continuous the dark green set intersect the geodesic. This implies that the distance from the grey point to the boundary upper bounds the distance to $\mathcal{S}$.

The following simple but very useful bound will be instrumental in the proof below.

▶ **Lemma 6.** *If a and b are two unit vectors we have that*

$$|\lambda a + (1-\lambda)b|^2 = 1 - (1-\lambda)\lambda|a-b|^2.$$

For vectors of arbitrary length we have

$$|\lambda a + (1-\lambda)b|^2 = \lambda|a|^2 + (1-\lambda)|b|^2 - \lambda(1-\lambda)|a-b|^2.$$

Proof. We have that for any two unit vectors a, b that:

$$\begin{aligned} |\lambda a + (1-\lambda)b|^2 &= \lambda^2|a|^2 + (1-\lambda)^2|b|^2 + 2\lambda(1-\lambda)a\cdot b && (2)\\ &= \lambda^2 + (1-\lambda)^2 + 2\lambda(1-\lambda)a\cdot b && \text{(using that } |a| = |b| = 1)\\ &= 1 - (1-\lambda)\lambda\,(2 - 2a\cdot b) \\ |a-b|^2 &= |a|^2 + |b|^2 - 2a\cdot b && (3)\\ &= 2 - 2a\cdot b && \text{(using that } |a| = |b| = 1) \end{aligned}$$

and thus

$$|\lambda a + (1-\lambda)b|^2 = 1 - (1-\lambda)\lambda|a-b|^2.$$

For a, b of arbitrary length we combine (2) and (3) as follows

$$\begin{aligned}|\lambda a + (1-\lambda)b|^2 &= \lambda^2|a|^2 + (1-\lambda)^2|b|^2 + \lambda(1-\lambda)\left(|a|^2 + |b|^2 - |a-b|^2\right)\\ &= \lambda|a|^2 + (1-\lambda)|b|^2 - \lambda(1-\lambda)|a-b|^2.\end{aligned}$$

◀

▶ **Theorem 1.** *Let $\mathcal{S}$ be a set of positive reach R, then for any two points $a, b \in \mathcal{S}$ such that the geodesic distance on $\mathcal{S}$ satisfies $d_{\mathcal{S}}(a,b) < \pi R$ there is a unique minimizing geodesic in $\mathcal{S}$ connecting a and b.*

Overview of the proof. The core idea of the proof is the following: Given two geodesics γ_1 and γ_2 connecting a, b that are parametrized by arc length we have that $\lambda\gamma_1 + (1-\lambda)\gamma_2$ is shorter than γ_1 and γ_2 because $|\lambda\dot{\gamma}_1 + (1-\lambda)\dot{\gamma}_2| < 1$, if $\lambda \in (0,1)$. So[2] if $\lambda\gamma_1 + (1-\lambda)\gamma_2 \subset \mathcal{S}$ we would find a contradiction, because there would be a curve connecting a and b that would be shorter than a geodesic. The main difficulty of the proof is that $\lambda\gamma_1 + (1-\lambda)\gamma_2$ is generally not in $\mathcal{S}$, but we need to project on $\mathcal{S}$ to find a curve that is. The technical difficulty lies with dealing with this projection.

The proof, perhaps surprisingly, also relies on a result from mathematical physics. Consider the Lagrangian of the (real) harmonic oscillator:

$$\mathcal{L}_{\mathrm{HO}} = |\dot{x}|^2 - \frac{|x|^2}{R^2}$$

Write S_{HO} for the action, that is

$$S_{\mathrm{HO}} = \int_0^\tau \mathcal{L}_{\mathrm{HO}} dt. \tag{4}$$

The Euler-Lagrange equation,

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{x}} = \frac{\partial \mathcal{L}}{\partial x},$$

in this case yields

$$\ddot{x}(t) = -\frac{x(t)}{R^2},$$

with the familiar solutions $c_1 \sin(t/R) + c_2 \cos(t/R) = c_4 \sin\left(\frac{t-t_0}{R}\right)$. We have the following, see [19, Section V], [28] for this specific result[3] and [17] for general theory:

▶ **Theorem 7** (Adjusted from [19] and [28])**.** *The solution to the Euler-Lagrange equation of the harmonic oscillator is a global minimum of the action* (4) *provided that $\tau < \pi R$.*

This means in particular that:

[2] Here abuse notation a little bit because we do not distinguish the notation for a geodesic and its parametrization.

[3] Note that our the Lagrangian of a high dimensional harmonic oscillator (with constant potential) is the sum of one-dimensional harmonic oscillators.

▶ **Corollary 8.** *Let c_4 be a constant. If $\tau < \pi R$ and $x(t)$ is a function that satisfies $x(0) = 0$ and $x(\tau) = c_4 \sin\left(\frac{\tau}{R}\right)$, then we have that*

$$\begin{aligned} S_{HO}(x) &= \int_0^\tau |\dot{x}|^2 - \frac{|x|^2}{R^2} dt \\ &\geq \int_0^\tau \left(\frac{c_4^2 \cos^2\left(\frac{t}{R}\right) - c_4^2 \sin^2\left(\frac{t}{R}\right)}{R^2} \right) dt \\ &= \int_0^\tau \frac{c_4^2}{R^2} \cos\left(\frac{2t}{R}\right) dt \\ &= \frac{c_4^2}{2R} \sin\left(\frac{2\tau}{R}\right). \end{aligned}$$

Proof of Theorem 1. Because $d_S(a,b) < \pi R$ the geodesic distance between a and b is finite, which means that there exists a path of finite length connecting a and b in $\mathcal{S}$. Moreover, thanks to the second paragraph of part III, section 1: "Die Existenz geodätischer Bogen in metrischen Räume" in [27], we have that: If there is at least one path between a and b with finite length, then a minimizing geodesic, i.e. a path with minimal length connecting a to b, exists. Therefore the set of minimizing geodesics is not empty.

Let $\gamma_1, \gamma_2 : [0, \ell] \to S$ be two length minimizing paths from a to b, parametrized by arc length, with same length: $\text{length}(\gamma_1) = \text{length}(\gamma_2) = \ell := d_S(a,b)$.

Denote by ϵ the maximal distance between $\gamma_1(t)$ and $\gamma_2(t)$, i.e.

$$\epsilon := \max_{t\in[0,\ell]} |\gamma_1(t) - \gamma_2(t)|, \tag{5}$$

and let $t_{\max} \in [0, \ell]$ be a parameter for this distance is attained, that is,

$$d\big(\gamma_1(t_{\max}), \gamma_2(t_{\max})\big) = \epsilon.$$

Without loss of generality, we assume that $t_{\max} \leq \ell/2$.

We now consider the two families of curves

$$\gamma_\lambda^{\mathbb{E}}(t) = \lambda\gamma_1(t) + (1-\lambda)\gamma_2(t)$$

and

$$\gamma_\lambda^{\mathcal{S}}(t) = \pi_{\mathcal{S}}(\gamma_\lambda^{\mathbb{E}}(t)).$$

We now upper bound the length $\gamma_\lambda^{\mathcal{S}}(t)$. By Lemma 4 we have that

$$d(\gamma_\lambda^{\mathbb{E}}(t), \mathcal{S}) \leq R - \sqrt{R^2 - \lambda(1-\lambda)|\gamma_1(t) - \gamma_2(t)|^2}.$$

Using Corollary 5 this yields that

$$\begin{aligned} &\text{length}(\gamma_\lambda^{\mathcal{S}}(t)) \\ &\leq \int_0^\ell \frac{1}{1 - (1 - \sqrt{1 - \lambda(1-\lambda)|\gamma_1(t) - \gamma_2(t)|^2/R^2})} |\lambda\dot{\gamma}_1(t) + (1-\lambda)\dot{\gamma}_2(t)| dt. \end{aligned} \tag{6}$$

We now focus on the second term in the product in the integral.

With Lemma 6 we find the following bound on (6):

$$\text{length}(\gamma_\lambda^{\mathcal{S}}(t)) \leq \int_0^\ell \frac{\sqrt{1 - (1-\lambda)\lambda(|\dot{\gamma}_1(t) - \dot{\gamma}_2(t)|^2)}}{\sqrt{1 - \lambda(1-\lambda)|\gamma_1(t) - \gamma_2(t)|^2/R^2}} dt. \tag{7}$$

Assuming that $(1-\lambda)\lambda$ is very small (more precisely we'll show that the length decreases in first order near $\lambda = 0$ or $\lambda = 1$) yields that

$$\begin{aligned}
&\int_0^\ell \frac{\sqrt{1-(1-\lambda)\lambda(|\dot\gamma_1(t)-\dot\gamma_2(t)|^2)}}{\sqrt{1-\lambda(1-\lambda)|\gamma_1(t)-\gamma_2(t)|^2/R^2}}dt. \\
&= \int_0^\ell \left(1+\lambda(1-\lambda)\frac{|\gamma_1(t)-\gamma_2(t)|^2}{2R^2}\right)\left(1-\frac{1}{2}(1-\lambda)\lambda(|\dot\gamma_1(t)-\dot\gamma_2(t)|^2)\right)dt \\
&\quad + \mathcal{O}\left(\lambda^2(1-\lambda)^2\right) \qquad \text{(by Taylor)} \\
&= \int_0^\ell 1+\lambda(1-\lambda)\frac{|\gamma_1(t)-\gamma_2(t)|^2}{2R^2} - \frac{1}{2}(1-\lambda)\lambda(|\dot\gamma_1(t)-\dot\gamma_2(t)|^2)dt \\
&\quad + \mathcal{O}\left(\lambda^2(1-\lambda)^2\right) \\
&= \int_0^\ell 1-\frac{1}{2}(1-\lambda)\lambda\left(|\dot\gamma_1(t)-\dot\gamma_2(t)|^2 - \frac{|\gamma_1(t)-\gamma_2(t)|^2}{R^2}\right)dt \\
&\quad + \mathcal{O}\left(\lambda^2(1-\lambda)^2\right). \qquad (8)
\end{aligned}$$

Now writing $\Delta(t) = \gamma_1(t) - \gamma_2(t)$ and thus $\dot\Delta(t) = \dot\gamma_1(t) - \dot\gamma_2(t)$, this integral (ignoring the higher order terms) takes the form of a familiar Lagrangian integral or action

$$\int_0^\ell \mathcal{L}\,dt = \int_0^\ell 1-\frac{1}{2}(1-\lambda)\lambda\left(|\dot\Delta(t)|^2 - \frac{|\Delta(t)|^2}{R^2}\right)dt, \qquad (9)$$

which we recognize (up to global additive and multiplicative factors) as the action integral of the harmonic oscillator. To achieve better bounds we'll now split this into two integrals

$$\int_0^\ell \mathcal{L}\,dt = \int_0^{\ell/2} \mathcal{L}\,dt + \int_{\ell/2}^\ell \mathcal{L}\,dt.$$

We'll only concentrate on the first integral, the second one can be treated in a completely symmetrical manner.

We now invoke Theorem 7 and Corollary 8, with $\tau = \ell/2$, together with (7), (8), and (9), to find that

$$\begin{aligned}
\text{length}(\gamma_\lambda^{\mathcal{S}}(t)) &\le \int_0^\ell 1-\frac{1}{2}(1-\lambda)\lambda\left(|\dot\gamma_1(t)-\dot\gamma_2(t)|^2 - \frac{|\gamma_1(t)-\gamma_2(t)|^2}{R^2}\right)dt \\
&\quad + \mathcal{O}\left(\lambda^2(1-\lambda)^2\right) \\
&= \left(\int_0^{\ell/2} + \int_{\ell/2}^\ell\right)\left(1-\frac{1}{2}(1-\lambda)\lambda\left(|\dot\gamma_1(t)-\dot\gamma_2(t)|^2 - \frac{|\gamma_1(t)-\gamma_2(t)|^2}{R^2}\right)\right)dt \\
&\quad + \mathcal{O}\left(\lambda^2(1-\lambda)^2\right) \\
&\le 2\left(\ell/2 - \frac{1}{2}(1-\lambda)\lambda\frac{c_4^2}{2R}\sin\left(\frac{\ell}{R}\right)\right) + \mathcal{O}\left(\lambda^2(1-\lambda)^2\right), \\
&\qquad \text{(the } \le \text{ follows because } -\tfrac{1}{2}(1-\lambda)\lambda \text{ is negative)}
\end{aligned}$$

where c_4 is some constant. This is strictly less than ℓ, as long as $\ell < \pi R$. This contradicts the assumption that the geodesics have length ℓ (which is the sum of the integral from 0 to $\ell/2$ and the integral from $\ell/2$ to ℓ), because now we found a curve that is shorter up to first order. ◀

2.3 Continuity

The uniqueness of the geodesics of length less than πR implies continuity for geodesics under the same restriction. The proof of the continuity will rely heavily on the Arzelà-Ascoli theorem, see e.g. [13]. To this end we have to recall some nomenclature: A sequence

$\{\gamma_n\}_{n\in\mathbb{Z}_{\geq 0}}$ of continuous functions on an interval $I \subset \mathbb{R}$ is uniformly bounded if for all n and all $t \in I$, we have that $|\gamma_n(x)| \leq C$, with C some (universal) constant. The sequence $\{\gamma_n\}_{n\in\mathbb{Z}_{\geq 0}}$ is uniformly equicontinuous if, for every $\epsilon > 0$ there exists a $\delta > 0$, such that for all t, t' satisfying $|t - t'| \leq \delta$ and all n we have that

$$|\gamma_n(t) - \gamma_n(t')| \leq \epsilon.$$

The theorem can now be stated as

▶ **Theorem 9** (Arzelà-Ascoli). *Let $\{\gamma_n\}_{n\in\mathbb{Z}_{\geq 0}}$ be a sequence of real-valued uniformly bounded, equicontinuous functions on a compact interval I. Then there exists a subsequence $\{\gamma_{n_k}\}_{k\in\mathbb{Z}_{\geq 0}}$ that converges uniformly.*

With these preliminaries out of the way we can state the second contribution of the paper:

▶ **Corollary 2.** *Let $\mathcal{S}$ be a set of positive reach R, then the geodesic connecting any two points $a, b \in \mathcal{S}$ is continuous (in the C^0 sense) with respect to the endpoints provided that $d_{\mathcal{S}}(a, b) < \pi R$.*

We provide two proofs, one here and another in the appendix. Although they share a fair number of common techniques they differ significantly on the way that they treat the length of the limit curve. Moreover, the proof here is shorter, but requires some familiarity with using the Arzelà-Ascoli theorem, while the proof in the appendix discusses this in greater detail.

Proof. Let a_n, b_n be a pair of sequences of points in $\mathcal{S}$, where $\lim_{n\to\infty} a_n = a$ and $\lim_{n\to\infty} b_n = b$, as well as a sequence of geodesics γ_n, connecting a_n and b_n. Let γ be the minimizing geodesic connecting a and b.

We want to prove that $\lim_{n\to\infty} \|\gamma - \gamma_n\|_{\sup} = 0$, where $\|\cdot\|_{\sup}$ denotes the C^0 norm (i.e. sup norm). In other words, we have to prove that, for any $\epsilon > 0$, the cardinal of the set:

$$\Gamma_\epsilon := \{\gamma_n \mid \|\gamma - \gamma_n\|_{\sup} \geq \epsilon\}$$

is finite.

Assume to achieve a contradiction that the cardinal of Γ_ϵ is infinite. Since the set of functions in Γ_ϵ is bounded and equicontinuous (because they admit a uniform Lipschitz constant), we can apply Arzelà-Ascoli theorem and find a subsequence $\{\gamma_{n_k}, k \in \mathbb{N}\} \subset \Gamma_\epsilon$ converging to a curve γ' in $\mathcal{S}$. Since $|\text{length}(\gamma) - \text{length}(\gamma_{n_k})| = |d_{\mathcal{S}}(a, b) - d_{\mathcal{S}}(a_{n_k}, b_{n_k})| \leq d_{\mathcal{S}}(a, a_{n_k}) + d_{\mathcal{S}}(b, b_{n_k})$.) one has $\lim_{k\to\infty} \text{length}(\gamma_{n_k}) = \text{length}(\gamma)$.

By a classical result [8, Claim 5.8, page 19], the length of the limit cannot be larger than the limit inf of lengths which yields $\text{length}(\gamma') \leq \lim_{k\to\infty} \text{length}(\gamma_{n_k}) = \text{length}(\gamma)$. Therefore, γ' would be a geodesic between a and b with $\|\gamma - \gamma'\|_{\sup} \geq \epsilon$ which contradicts the unicity of geodesics stated in Theorem 1. ◀

Corollary 2 has itself another consequence which is worth pointing out.

▶ **Corollary 10.** *A geodesic ball of radius less than πR is contractible.*

Proof. Write $B_{\mathcal{S}}(c, r)$, for the geodesic ball in $\mathcal{S}$ with centre c and radius r, where $r < \pi R$. Let $x \in B_{\mathcal{S}}(c, r)$, and let $\gamma_{xc}(t)$ be the geodesic parametrized by $[0, 1]$ with constant arclength with $\gamma_{xc}(1) = x$ and $\gamma_{xc}(0) = c$. Thanks to Corollary 2 the geodesics are continuous under the sup-norm, with respect to endpoint x. This means that the map $H : [0, 1] \times B_{\mathcal{S}}(c, r) : (\lambda, x) \mapsto \gamma_{xc}(\lambda)$ is a (continuous) homotopy. ◀

References

1 Eddie Aamari, Clément Berenfeld, and Clément Levrard. Optimal reach estimation and metric learning. *The Annals of Statistics*, 51(3):1086–1108, 2023.

2 Eddie Aamari and Clément Levrard. Stability and minimax optimality of tangential Delaunay complexes for manifold reconstruction. *Discrete & Computational Geometry*, 59(4):923–971, 2018. doi:10.1007/s00454-017-9962-z.

3 N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, December 1999. doi:10.1007/PL00009475.

4 Ery Arias-Castro and Bruno Pelletier. On the convergence of maximum variance unfolding. *The Journal of Machine Learning Research*, 14(1):1747–1770, 2013. doi:10.5555/2567709.2567719.

5 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and Topological Inference*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2018. doi:10.1017/9781108297806.

6 Jean-Daniel Boissonnat, André Lieutier, and Mathijs Wintraecken. The reach, metric distortion, geodesic convexity and the variation of tangent spaces. *Journal of Applied and Computational Topology*, 3(1):29–58, June 2019. doi:10.1007/s41468-019-00029-8.

7 Martin R Bridson and André Haefliger. *Metric spaces of non-positive curvature*, volume 319 of *Grundlehren der mathematischen Wissenschaften*. Springer Science & Business Media, 2013.

8 H. Busemann. *The geometry of geodesics*. Dover Publications, 2005.

9 Ratthachat Chatpatanasiri, Teesid Korsrilabutr, Pasakorn Tangchanachaianan, and Boonserm Kijsirikul. A new kernelization framework for Mahalanobis distance learning algorithms. *Neurocomputing*, 73(10-12):1570–1579, 2010. doi:10.1016/J.NEUCOM.2009.11.037.

10 S.-W. Cheng, T.K. Dey, and J.R. Shewchuk. *Delaunay Mesh Generation*. Computer and information science series. CRC Press, 2013.

11 Michael AA Cox and Trevor F Cox. Multidimensional scaling. In *Handbook of data visualization*, pages 315–347. Springer, 2008.

12 T.K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Number 23 in Cambridge monographs on applied and computational mathematics. Cambridge University Press, 2007.

13 Nelson Dunford and Jacob T Schwartz. *Linear operators, part 1: general theory*. John Wiley & Sons, 1988.

14 H. Federer. Curvature measures. *Transactions of the American Mathematical Society*, 93:418–491, 1959.

15 Charles Fefferman, Sergei Ivanov, Yaroslav Kurylev, Matti Lassas, and Hariharan Narayanan. Reconstruction and interpolation of manifolds. I: The geometric Whitney problem. *Foundations of Computational Mathematics*, 2019. doi:10.1007/s10208-019-09439-7.

16 Charles Fefferman, Sergei Ivanov, Matti Lassas, and Hariharan Narayanan. Fitting a manifold of large reach to noisy data. *arXiv e-prints*, page arXiv:1910.05084, October 2019. arXiv:1910.05084.

17 Izrail Moiseevitch Gelfand and Richard A Silverman. *Calculus of variations*. Courier Corporation, 2000.

18 Amir Globerson and Sam T. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

19 C. G. Gray and Edwin F. Taylor. When action is not least. *American Journal of Physics*, 75(5):434–458, May 2007. doi:10.1119/1.2710480.

20 Mikhael Gromov. Hyperbolic groups. In *Essays in group theory*, pages 75–263. Springer, 1987.

21 Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? Metric learning approaches for face identification. In *2009 IEEE 12th International Conference on Computer Vision*, pages 498–505. IEEE, 2009. doi:10.1109/ICCV.2009.5459197.

22 Geoffrey Hinton and Sam T. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, volume 15, 2002.

23 James T. Kwok and Ivor W. Tsang. Learning with idealized kernels. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 400–407, 2003. URL: `http://www.aaai.org/Library/ICML/2003/icml03-054.php`.

24 André Lieutier and Mathijs Wintraecken. Manifolds of positive reach, differentiability, tangent variation, and attaining the reach. In Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri, editors, *42nd International Symposium on Computational Geometry (SoCG 2026)*, volume 367 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 75:1–75:16, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

25 Alexander Lytchak. On the geometry of subsets of positive reach. *Manuscripta mathematica*, 115(2):199–205, 2004.

26 Alexander Lytchak. Almost convex subsets. *Geometriae Dedicata*, 115(1):201–218, 2005.

27 K. Menger. Untersuchungen über allgemeine Metrik. Vierte Untersuchung. Zur Metrik der Kurven. *Mathematische Annalen*, 103:466–501, 1930.

28 M. Moriconi. Condition for minimal harmonic oscillator action. *American Journal of Physics*, 85(8):633–634, August 2017. `doi:10.1119/1.4984778`.

29 Bac Nguyen, Carlos Morell, and Bernard De Baets. Supervised distance metric learning through maximization of the Jeffrey divergence. *Pattern Recognition*, 64:215–225, 2017. `doi:10.1016/J.PATCOG.2016.11.010`.

30 Barak Sober and David Levin. Manifold approximation by moving least-squares projection (MMLS). *Constructive Approximation*, pages 1–46, 2019. `doi:10.1007/s00365-019-09489-8`.

31 M. Spivak. *A comprehensive introduction to differential geometry: Volume I*. Publish or Perish, 1999.

32 José L. Suárez, Salvador García, and Francisco Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing*, 425:300–322, 2021. `doi:10.1016/J.NEUCOM.2020.08.017`.

33 Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

34 Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart J. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, volume 15, 2002.

35 Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. Technical Report MSU-CSE-06-2, Michigan State University, 2006.

A Alternative proof

Alternative proof of Corollary 2. Assume that the geodesics are not continuous with respect to the endpoints, then there exist an $\epsilon > 0$ and a pair of sequences of points a_n, b_n, where $\lim_{n\to\infty} a_n = a$ and $\lim_{n\to\infty} b_n = b$, as well as a sequence of geodesics γ_n, connecting a_n and b_n such that

$$\sup_{t\in I} |\gamma_n(t) - \gamma(t)| \geq \epsilon,$$

where γ is the geodesic connecting a and b. We stress that here we parametrize the geodesics on the unit interval $[0, 1]$, with constant norm of the tangent vector, i.e. $|\dot{\gamma}_n| = d_{\mathcal{S}}(a_n, b_n)$.

We further note that thanks to the metric characterization of sets of positive reach [6, Theorem 1] convergence in the Euclidean and geodesic sense are equivalent.

Again, thanks to the metric characterization of sets of positive reach [6, Theorem 1] we know that geodesics are themselves sets of (at least) the same reach as the set in which they lie. We also know that manifolds of positive reach are $C^{1,1}$ where the Lipschitz constant of the derivative is bounded by $1/R$, see [24]. This implies that the tangent vectors $\dot{\gamma}_n$ are uniformly Lipschitz, that is Lipschitz with the same Lipschitz constant for all n. This

uniform Lipschitz constant implies by definition uniform equicontinuity of $\dot{\gamma}_n$. Because all geodesics are parametrized according to arclength, with $|\dot{\gamma}_n| = d_{\mathcal{S}}(a_n, b_n)$, and $d_{\mathcal{S}}(a_n, b_n)$ is bounded by assumption, $\dot{\gamma}_n$ is also bounded.

We now observe that by passing to a subsequence we can assume that there is one $t^* \in I$ such that

$$|\gamma_n(t^*) - \gamma(t^*)| \geq \epsilon/2. \tag{10}$$

This can be seen as follows: Because I is compact the supremum must be attained at some point t_n^* for each $(\gamma_n - \gamma)(t)$. Again because I is compact this sequence of points must have a convergent subsequence n_m with limit t^* because of smoothness (more precisely, the fact that $|\dot{\gamma}_n|$ is bounded) of $(\gamma_n)(t)$, we have that for m sufficiently large and $|t - t^*|$ sufficiently small, we have $|\gamma_{n_m}(t) - \gamma(t)| \geq \epsilon/2$. This implies that (10) holds.

Now we can invoke the Arzelà-Ascoli theorem to conclude that there exist a $\dot{\gamma}'$, which is the limit of a convergent subsequence of $\dot{\gamma}_n$. We write $\gamma'(t) = \int_0^t \dot{\gamma}' \mathrm{d}s$. With this notation we can formulate the following observations:

- The lengths of γ_n converge to the length of γ'.
- The geodesic γ_n has to converge to the geodesic distance between a and b, because by the triangle inequality we have that

  $$|\text{length}(\gamma) - \text{length}(\gamma_n)| = |d_{\mathcal{S}}(a, b) - d_{\mathcal{S}}(a_n, b_n)| \leq d_{\mathcal{S}}(a, a_n) + d_{\mathcal{S}}(b, b_n).$$

  And hence, γ and γ' both are geodesics, that is their lengths equal the geodesic distance between the endpoints.
- $|\gamma(t^*) - \gamma'(t^*)| \geq \epsilon/2$, in particular $\gamma \neq \gamma'$.

This yields a contradiction with the uniqueness of the geodesics, that is Theorem 1. ◀

A Fast Algorithm for the Hecke Representation of the Braid Group, and Applications to the Computation of the HOMFLY-PT Polynomial and the Search for Interesting Braids

Clément Maria
Inria d'Université Côte d'Azur, Valbonne, France

Hoel Queffelec
France-Australia Mathematical Sciences and Interactions ANU - CNRS International Research Laboratory, Canberra, Australia

Abstract

Knot theory is an active field of mathematics, in which combinatorial and computational methods play an important role. One side of computational knot theory, that has gained interest in recent years, both for complexity analysis and practical algorithms, is quantum topology and the computation of topological invariants issued from the theory.

In this article, we leverage the rigidity brought by the representation-theoretic origins of the quantum invariants for algorithmic purposes. We do so by exploiting braids and the algebraic properties of the braid group to describe, analyze, and implement a fast algorithm to compute the Hecke representation of the braid group. We apply this construction to design a parameterized algorithm to compute the HOMFLY-PT polynomial of knots, and demonstrate its interest experimentally. Finally, we combine our fast Hecke representation algorithm with Garside theory, to implement a reservoir sampling search and find non-trivial braids with trivial Hecke representations with coefficients in $\mathbb{Z}/p\mathbb{Z}$. We find explicitly several such braids, for the 4-strand and 5-strand braid groups.

2012 ACM Subject Classification Mathematics of computing → Geometric topology; Theory of computation → Fixed parameter tractability

Keywords and phrases Hecke representation of the braid group, parameterized algorithm, HOMFLY-PT polynomial of knots, reservoir sampling, faithfulness of Hecke representation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.76

Related Version *Full Version*: `https://arxiv.org/abs/2512.06142` [29]

Supplementary Material *Software (Source code)*: `https://doi.org/10.5281/zenodo.19349647`

Funding *Clément Maria*: Partially supported by the ANR project ANR-20-CE48-0007 (AlgoKnot).

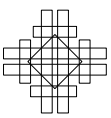
Acknowledgements We would like to thank Asilata Bapat, Stepan Orevkov – who pointed us to Morton's 1985 implementation [31] of the HOMFLY-PT algorithm – , Emmanuel Wagner and Oded Yacobi for helpful discussions. We would also like to thank the anonymous referees for the detailed comments and useful feedback. Some experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see `https://www.grid5000.fr`).

1 Introduction

Geometrically, a *braid* on n strands is the embedding of n non-intersecting paths in the 3-dimensional space $\mathbb{R}^2 \times [0,1]$, such that every path connects a point $(i,0,0), i \in \{1,\dots,n\}$ in the bottom plane to a point $(j,0,1), j \in \{1,\dots,n\}$ in the top plane, and every path grows monotonically along the z-axis. Two braids are *equivalent* if there is an ambient isotopy of



© Clément Maria and Hoel Queffelec;
licensed under Creative Commons License CC-BY 4.0



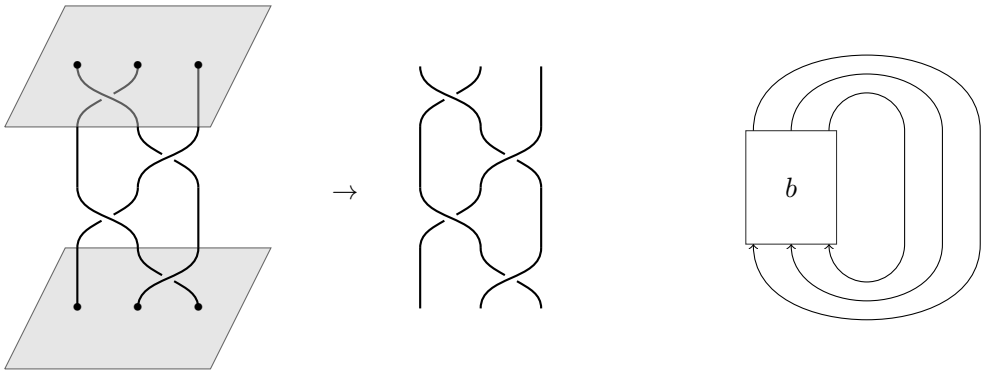


Figure 1 Left: Geometric braid on 3 strands in $\mathbb{R}^2 \times [0;1]$, and natural diagram (projection on xz-plane). Right: Braid closure $\widehat{b}$ of braid b.

$\mathbb{R}^2 \times [0,1]$ fixing the bottom and top planes and taking one braid to the other. Braids are generally represented by *braid diagrams*, that are planar projections of a braid along the y-axis, keeping track of upper and under crossings; see Figure 1.

Braids are notably important in knot theory, as any link can be represented as the closure of a braid [1]. Knots have been studied extensively under the algorithmic lens. A famous problem is the computational complexity of recognizing the *trivial knot* from an input diagram, which is known to be in the complexity classes **NP** [19] and **coNP** [24], for which the best known worst case algorithm is exponential [19], but which experimentally exhibits a fast polynomial time behavior [11] with optimized implementation. In particular, the experimental aspects of computational knot theory play a fundamental role in the field, where mathematicians and computer scientists use efficient software, such as `Regina` [6, 9] and `SnapPy` [12], as well as computer-constructed census of knots [8], to guess and challenge profound conjectures, *e.g.*[5, 14, 16]. Consequently, an important side of computational knot theory is the design and implementation of fast, highly optimized algorithms.

Motivated by a finer understanding of the complexity of problems related to knots as well as practically fast computation, a recent route of research uses tools from *parameterized complexity* to compute *topological invariants* of knots. This approach has been particularly successful for invariants constructed via *quantum topology*, a field of topology using tools from quantum mechanics. In particular, the *treewidth* of a graph is a parameter measuring how close the graph is to a tree and, similarly, the *pathwidth* measures the proximity to a path. They can be extended to knot theory by considering the graph obtained from a diagram by putting a vertex on each crossing and an edge for each strand connecting crossings.

These are important parameters capturing a certain notion of sparsity of the input, and they can be combined with algorithmic techniques such as *dynamic programming* to design algorithms whose complexity depends exponentially on the tree/pathwidth and only polynomially in the size of the input: in quantum topology, such algorithms have been designed for the Jones and Kauffman polynomials [27], the Reshetikhin-Turaev invariants [28], and the HOMFLY-PT polynomial [7], which are **#P-hard** to compute in general [23]. Note that these quantum invariants, together with theoretically and practically fast algorithms, have been applied to construct knot censuses [8].

Contrary to knots, braids on n-strands have a natural algebraic description as a group, the *braid group* B_n, yielding rich algebraic properties that we exploit heavily in the following.

Our results. The goal of this article is to describe fast algorithms and data structures for braids and the braid group, related to quantum topology, and to apply them to computational knot theory and experimental mathematics. As opposed to algorithms on knots mentioned above, these algorithms rely heavily on the algebraic structure of the braid group.

Our starting point is to consider the *Hecke representation* of a braid. The *Hecke algebra* $\mathcal{H}_n$ is a fundamental concept in modern mathematics [30], deeply connected to group theory, number theory, and knot theory. The braid group B_n admits a representation into the Hecke algebra $\mathcal{H}_n$, *i.e.*, a map $B_n \to \mathcal{H}$ respecting the group structure. Knowing whether this map (with $\mathbb{Z}$-coefficients) is *faithful* (*i.e.*, only the trivial braid has a trivial image) is a major open question [35, 21, 25, 4], related to the detection of the unknot by the Jones polynomial [20].

Our algorithm scans an input braid diagram from bottom to top, and updates its representation in a basis of $\mathcal{H}$. This strategy was already used by Morton and Short in the 80's [32]; the program we wrote can be seen as an optimized and parallelized version of theirs.

▶ **Theorem 1** (see Theorem 10). *Given a braid $b \in B_n$ on n strands and with N crossings, there is an algorithm to compute its Hecke representation in $O(n!(n \log n) \times N)$ operations and $O(n! \times N)$ algebraic operations, storing at most $n! + O(1)$ algebraic elements over the computation.*

Every knot or link can be obtained from the braid closure of a braid [1]; see Figure 1. Additionally, the HOMFLY-PT polynomial of the knot/link obtained from the braid closure can be computed from the Hecke representation of the braid by taking a *trace*. This is an alternative, more algebraic approach, to the usual definition of quantum invariants used in parameterized algorithms in the literature, relying either on *state sums* [10], *tensor networks* [28], or skein relations [7]. Here, we exploit the rigidity provided by the algebraic or representation-theoretic point of view to design more efficient algorithmic processes. In particular, the morphism spaces between representations are of much lower dimension than those between the underlying vector spaces, which helps reducing the algorithmic complexity.

We describe a fast implementation of the trace operation in the single pass of the coordinate vector representing the Hecke element associated to the braid.

▶ **Theorem 2** (see Theorem 19). *Given a braid $b \in B_n$ with m crossings, there is an algorithm to compute its HOMFLY-PT polynomial with the same complexity as Theorem 1.*

We compare experimentally this algorithm against Burton's implementation [7, 9] on large families of braid closures, and demonstrate its practical interest.

Finally, we use these fast implementations in experimental mathematics, running an extensive search for counter-examples to the Hecke faithfulness question in the case of Hecke representation with $\mathbb{Z}/p\mathbb{Z}$ coefficients.

Taking inspiration from earlier works [2, 3, 18], we have implemented a random algorithm to find counter-examples to faithfulness in the braid groups B_4 and B_5, for different coefficients $\mathbb{Z}/p\mathbb{Z}$. Exploiting the favorable algebraic properties of braids, we have used *Garside theory* to generate increasingly complicated, non-trivial, braids for the search, running on each of them the algorithm from Theorem 10 in order to find braids whose Hecke representation was getting increasingly close to the trivial one. We have found explicit non-trivial braids in B_4 whose Hecke image is trivial in coefficients $\mathbb{Z}/2\mathbb{Z}$, $\mathbb{Z}/3\mathbb{Z}$ and $\mathbb{Z}/4\mathbb{Z}$. The non-faithfulness of the Hecke representation was known abstractly in these cases, using the explicit decomposition into simpler representations [21, Sections 5.7, 8.5], but our algorithm provides an efficient way to find examples of elements in the kernel. Similarly, we have explicitly found a non-trivial braid in B_5 whose Hecke image modulo 2 is trivial, while the status of the faithfulness of the Hecke representation of B_5 only relies on embedding B_4 into B_5.

The code is available as part of the `KumQuAT` library [36].

Comparison with the literature. The HOMFLY-PT polynomial is a powerful topological invariants, and its computation has attracted the attention of mathematicians. For a knot diagram with N crossings, Kauffman gave a fast $O(2^N)$ skein template algorithm [22]. More recently, starting from Kauffman's work, Burton [7] designed the first fixed parameter tractable algorithm in the treewidth of the knot diagram, running in $O((2\,\mathrm{tw})!^4 \cdot \mathrm{poly}(N))$, and implemented it in the software `Regina` [9]. This complexity bound should be compared to our bound from Theorem 1.

While treewidth, over all possible diagrams of a knot, is a generally smaller parameter than the braid index, it is common for them to be essentially equal, and the diagrams minimizing the treewidth are braid closures ; this is the case for *torus knots* for example [34, 26]. On the other hand, the complexity dependence of our algorithm in the number of strands is much lower than Burton's dependence in the treewidth, which proves an advantage in practice on particular families of examples where treewidth and braid index are close.

Our approach to find counter-examples to the faithfulness of the Hecke representation is inspired from recent works [18] in computational mathematics, which have used a combination of curve-based random search with Garside theory to study the faithfulness of the *Burau representation* of the braid group, and in particular to find braids with trivial representations. However, the Burau representation of a braid is significantly less costly to compute, with a (small) polynomial dependence in both number of strands and number of crossings, while all known algorithms for the HOMFLY-PT polynomial, including this work, have a super-exponential dependence in the number of strands.

2 Preliminaries

2.1 Permutations and compact encoding

The *symmetric group* Σ_n is the group of permutations of n elements $\{1, \dots, n\}$. The group Σ_n is generated by the *transpositions*, *i.e.*, the permutations s_i of the two consecutive elements at index i and $i+1$, for $1 \leq i \leq n-1$. Every permutation $w \in \Sigma_n$ can be represented either by a unique *one-line word* $w(1)w(2)\cdots w(n)$, where $w(i)$ is the image of i under the permutation w, or by a non-unique product of generators $w = s_{i_k} \cdots s_{i_1}$.

An *inversion* in a permutation w is a pair of indices (i, j) such that $i < j$ and $w(i) > w(j)$. The *(Coxeter) length* of a permutation w, denoted by $\ell(w)$, is its total number of inversions.

The *Lehmer code* $L(w)$ of a permutation $w = w(1)w(2)\cdots w(n)$ is the sequence of positive integers $L(w) := (L(w)_1, \dots, L(w)_n)$ such that:

$$L(w)_i := \#\{j > i : w(j) < w(i)\},$$

i.e., $L(w)_i$ counts the number of inversions happening to the right of index i. In particular, $0 \leq L(w)_i \leq n - i$. Permutations are in bijection with their Lehmer code.

This last property allows us to represent a permutation with a single number, using the *factorial number system*. Specifically, given a sequence of numbers $\mathbf{a} = (a_1, \dots, a_n)$ such that $0 \leq a_i \leq n - i$ for all i, one can write uniquely the sequence as a single positive integer:

$$(a_1, \dots, a_n) \to F_{\mathbf{a}} = \sum_{i=1}^{n} a_i \times (n-i)! \tag{1}$$

In particular, the application $\Sigma \to \{0, \dots, n! - 1\}$ sending $w \mapsto F_{L(w)}$ is a bijection. In the following, we call the number $F_{L(w)}$ the *index of the permutation* w.

We prove, in the long version [29], the following basic facts about the encoding:

▶ **Lemma 3.**

(i) *Given an index $F_{L(w)} \in \{0, \dots, n!-1\}$, there is an $O(n \log n)$ time algorithm to compute a one-line word presentation of w,*

(ii) *Given a permutation w represented as a one-line word $w(1) \cdots w(n)$, there is an $O(n \log n)$ time algorithm to compute its index $F_{L(w)}$,*

(iii) *Let $w, w' \in \Sigma_n$ be permutations, then:*

$$w(1) \cdots w(n) \leq_{\text{lex}} w'(1) \cdots w'(n) \text{ iff } F_{L(w)} \leq F_{L(w')},$$

where $\leq_{\text{lex}}$ denotes the lexicographic order.

2.2 Braids and Garside structure

We consider B_n the braid group on n strands:

$$B_n = \left\langle \sigma_1, \dots, \sigma_{n-1} \;\middle|\; \begin{cases} \sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1} \\ \sigma_i \sigma_j = \sigma_j \sigma_i \text{ if } |i-j| > 1 \end{cases} \right\rangle.$$

The generators $\{\sigma_i^{\pm 1}\}_i$ are called the *Artin generators* of the braid group. The monoid with the same presentation, *i.e.*, generated by the $\{\sigma_i^{+1}\}_i$, is denoted by B_n^+.

For some of our applications, we are interested in producing braids that we want to make sure get more and more complicated. The notion of Garside structure is instrumental, by providing us with an algorithmic-friendly normal form. We very briefly review the most important aspects, but refer to [13] for details.

Let us denote $[1, \Delta]$ the braids that realize positive lifts (that is, in B_n^+) of reduced words from the symmetric group. For example, $s_1 s_2 s_1 = s_2 s_1 s_2 \in \Sigma_n$ lifts to $\sigma_1 \sigma_2 \sigma_1 = \sigma_2 \sigma_1 \sigma_2 \in B_n$. It is a classical result that the map is well-defined (independent on the choice for the starting word). The half-twist $\Delta = \sigma_1 \cdots \sigma_{n-1} \sigma_1 \cdots \sigma_{n-2} \cdots \sigma_1$ is the so-called *Garside element*. One has a lattice structure [17, 13] on $[1, \Delta]$, extending to B_n^+, with left divisibility given by:

$$b_1 \leq_L b_2 \;\Leftrightarrow\; \exists b_3 \in B_n^+,\; b_1 b_3 = b_2.$$

Similarly, one can define a notion of right divisibility, $\leq_R$. We are now ready to obtain the normal form $b_k \cdots b_1$ of a braid $b \in B_n^+$ by inductively defining:

$$b_i := \sup\{b' \in [1, \Delta],\; b' \leq_R (b b_1^{-1} \cdots b_{i-1}^{-1})\}.$$

If one adds the fact that any braid $b \in B_n$ can be written as $\Delta^r b^+$ with $b^+ \in B_n^+$, then one has a preferred expression for any braid.

At some point, we will be interested in using the Garside normal form to generate complicated braids. This will be made algorithmic thanks to the following lemma.

▶ **Lemma 4.** *The word $b_k \cdots b_1$ with $b_i \in [1, \Delta]$ is under normal form if and only if the following condition holds:*

$$\forall i \geq 2, \forall j, \quad \sigma_j \leq_R b_i \Rightarrow \sigma_j \leq_L b_{i-1}.$$

The divisibility in $[1, \Delta]$ being controlled by Σ_n, the computations can be made in the symmetric group. More details about the underlying automaton appear in Section 5.3.

2.3 Hecke algebra

The Hecke algebra is an ubiquitous object in representation theory. We introduce it as a quotient of the braid group, which makes it most natural for the computations we care about, and we refer to [29] and to Mathas' book [30] for further details and context.

▶ **Definition 5.** *Let $\mathcal{H}_n$ be the $\mathbb{Z}[q^{\pm 1}]$ module obtained from the group algebra of the braid group as follows:*

$$\mathcal{H}_n = \frac{\mathbb{Z}[q^{\pm 1}][B_n]}{\langle (\sigma_i - 1)(\sigma_i + q^2) \rangle}$$

We denote by $\psi : B_n \to \mathcal{H}_n$ the natural induced map. We quickly review some useful basic properties of the Hecke algebra that will be instrumental to us.

▶ **Lemma 6.** *Denote by T_{s_i} the image of the i-th generator $\sigma_i \in B_n$. Then there is a* well-defined *map (of sets):*

$$\Sigma_n \to \mathcal{H}_n \qquad\qquad s = s_{i_1} \cdots s_{i_k} \mapsto T_s := T_{s_{i_1}} \cdots T_{s_{i_k}}.$$

Furthermore the image $\{T_s\}_{s \in \Sigma_n}$ is a basis of the Hecke algebra as a $\mathbb{Z}[q^{\pm 1}]$-module.

The content of the above lemma is that the image of s does not depend on the word chosen to write it. This is simply because the T_s satisfy the same braiding relations as the elements in the symmetric group. To our eyes, the key property is that we have a basis indexed by permutations, which can be efficiently handled. For computation purposes, we also take advantage of the fact that the structure constants of this basis are quite simple, as expressed in the following classical lemmas [30, Theorem 1.11].

▶ **Lemma 7.**

$$\forall w \in \Sigma_n,\ \forall i,\ T_{s_i} T_w = \begin{cases} T_{s_i w} & \text{if} \quad \ell(s_i w) > \ell(w); \\ (1-q^2)T_w + q^2 T_{s_i w} & \text{if} \quad \ell(s_i w) < \ell(w). \end{cases}$$

▶ **Lemma 8.**

$$\forall w \in \Sigma_n,\ \forall i,\ T_{s_i}^{-1} T_w = \begin{cases} (1-q^{-2})T_w + q^{-2} T_{s_i w} & \text{if} \quad \ell(s_i w) > \ell(w); \\ T_{s_i w} & \text{if} \quad \ell(s_i w) < \ell(w). \end{cases}$$

When we compute knot invariants, it is convenient to take braid closures of braids. Assuming that P is a knot invariant, it is clear that $P(\widehat{b_1 b_2}) = P(\widehat{b_2 b_1})$. This remark motivates the introduction of an intermediary object $\mathrm{Tr}(\mathcal{H}_n)$ which is obtained from $\mathcal{H}_n$ by formally adding the relation $b_1 b_2 = b_2 b_1$. This makes it possible to break the computation into a first step $\mathcal{H}_n \to \mathrm{Tr}(\mathcal{H}_n)$, where elements are reduced as much as possible using the trace relation $b_1 b_2 = b_2 b_1$, before using information about the specific invariant we want to compute to go from $\mathrm{Tr}(\mathcal{H}_n)$ to $\mathbb{Z}[q^{\pm 1}]$. Note that because $\psi \colon B_n \to \mathcal{H}_n$ is a representation, for any two braids $b_1, b_2 \in B_n$, we have $\mathrm{Tr}(\psi(b_1 b_2) = \mathrm{Tr}(\psi(b_1)\psi(b_2)) = \mathrm{Tr}(\psi(b_2)\psi(b_1)) = \mathrm{Tr}(\psi(b_2 b_1))$.

Finally, when searching for counterexamples, we measure how far an element of $\mathcal{H}_n$ is from the identity. We use the following notion, that we adapt from [18].

▶ **Definition 9.** *Let $x = \sum_w a_w T_w \in \mathcal{H}_n$. We define its* projective length*:*

$$\mathrm{projlength}(x) = \max\{\deg(a_w),\ w \in \Sigma_n\} - \min\{\mathrm{val}(a_w),\ w \in \Sigma_n\},$$

where deg *and* val *are the degree and valuation of a Laurent polynomial in the variable q.*

Most of the notions we have defined here enjoy similar definitions with $\mathbb{Z}/p\mathbb{Z}$ in place of $\mathbb{Z}$. We use this context in Section 5.

3 Computing the Hecke representation of the braid group

The goal of the algorithm is to compute the image $\psi(b)$ in the Hecke algebra $\mathcal{H}_n$ of a braid $b = \sigma_{i_c}^{\epsilon_c} \cdots \sigma_{i_1}^{\epsilon_1} \in B_n$, given as an expression of the braid in the generators $\{\sigma_i\}$ of B_n.

Our algorithm is iterative, and maintains at step j the image in $\mathcal{H}_n$ of the braid $b_{(j)} := \sigma_{i_j}^{\epsilon_j} \cdots \sigma_{i_1}^{\epsilon_1}$, made of the first j crossings of the input braid.

3.1 Data structure and initialization

For a braid in B_n, we represent its image in $\mathcal{H}_n$ by an array of coordinates (Laurent polynomials) in the basis $\{T_w\}$. Specifically, a vector $x = \sum_{w\in\Sigma_n} a_w T_w \in \mathcal{H}_n$, with $a_w \in \mathbb{Z}[q^{\pm 1}]$, is represented by a length $n!$ array `hecke`, such that `hecke`$[j] = a_w$ whenever the permutation w has index $F_{L(w)} = j$.

Recall that the one-line word $w(1)\cdots w(n)$ of a permutation w, and its index $F_{L(w)}$ can be obtained from one another in $O(n \log n)$ operations. Note also that the maximal value $n! - 1$ of an index is below 2^{64} when $n \leq 20$, which will always be the case in this article ; permutation indices are stored as standard integers.

The image of σ_{i_1} is $T_{s_{i_1}} \in \mathcal{H}_n$, and Lemma 8, shows that the image of $\sigma_{i_1}^{-1}$ is $q^{-2}T_{s_{i_1}} + (1 - q^{-2}) \in \mathcal{H}_n$. Consequently, if $\epsilon_1 = 1$, at the initialization, only the coefficient corresponding to s_{i_1} is 1, all other ones being 0. If $\epsilon_1 = -1$, then we have the coefficient $1 - q^{-2}$ at index 0, q^{-2} at the index representing s_{i_1}, and 0 elsewhere.

3.2 Inductive step

We suppose that at the beginning of step j of the algorithm, `hecke` stores the coordinates of the image of $\sigma_{i_{j-1}}^{\epsilon_{j-1}} \cdots \sigma_{i_1}^{\epsilon_1}$, that we update with the next generator $\sigma_{i_j}^{\epsilon_j}$.

Since ψ is a group homomorphism, we have $\psi(b_{(j)}) = T_{\sigma_{i_j}}^{\epsilon_j} \psi(b_{(j-1)})$. We consequently need to update the coordinates of `hecke` following the local relations described in Lemmas 7 and 8. We do so in the single pass through `hecke`, without allocating more than $O(1)$ Laurent polynomials in memory:

Algorithm 1 From `hecke` representing $\psi(b_{(j-1)})$ to representing $\psi(b_{(j)})$.

for $i \leftarrow 0$ **to** $n! - 1$ **do**
  compute w such that $F_{L(w)} = i$;
  if $\ell(s_{i_j} w) > \ell(w)$ **then**
    set $\text{idx}_w \leftarrow F_{L(w)}$; $\text{idx}_{s_{i_j} w} \leftarrow F_{L(s_{i_j} w)}$; $a_w \leftarrow$ `hecke`$[\text{idx}_w]$;
     $a_{s_{i_j} w} \leftarrow$ `hecke`$[\text{idx}_w]$;
    if $\epsilon_j = +1$ **then**
      set `hecke`$[\text{idx}_w] \leftarrow q^2 a_{s_{i_j} w}$; set `hecke`$[\text{idx}_{s_{i_j} w}] \leftarrow a_w + (1 - q^2) a_{s_{i_j} w}$;
    end
    else
      set `hecke`$[\text{idx}_w] \leftarrow (1 - q^{-2}) a_w + a_{s_{i_j} w}$; set `hecke`$[\text{idx}_{s_{i_j} w}] \leftarrow q^{-2} a_w$;
    end
  end
end

3.3 Correctness

The correctness of the initialization phase is guaranteed by Lemmas 7 and 8. We focus on the inductive step. Consider any permutation $w \in \Sigma$. According to Lemmas 7 and 8, both $T_{s_{i_j}}^{\pm 1} T_w$ and $T_{s_{i_j}}^{\pm 1} T_{s_{i_j} w}$ give linear combinations of T_w and $T_{s_{i_j} w}$, because $s_{i_j}(s_{i_j} w) = w$. Additionally, for any permutation $w' \notin \{w, s_{i_j} w\}$, the product $T_{s_{i_j}}^{\pm 1} T_{w'}$ contains no term T_w or $T_{s_{i_j} w}$. Thus the pair of coordinates of `hecke` corresponding to permutations w and $s_{i_j} w$ can be updated independently from the rest of the array, and their new values is a combination of their previous values. We update them at once in the core of the (outer) `if` loop of Algorithm 1.

Finally, the condition of the (outer) `if` loop guarantees that we proceed to the update once. Indeed, the length ℓ being the number of inversions, we have $\ell(s_{i_j} w) \in \{\ell(w) - 1, \ell(w) + 1\}$. In consequence, considering the two permutations w and $(s_{i_j} w)$, multiplying by s_{i_j} exactly increases the length once and decreases the length once.

In light of the above, the induction algorithm does compute $\psi(b_{(j)})$ from $\psi(b_{(j-1)})$ and, at the end of the algorithm, we are left with $\psi(b) = \sum_{w \in \Sigma_n} a_w T_w$.

3.4 Complexity

▶ **Theorem 10.** *Given $b \in B_n$ represented by N generators (N crossings), the algorithm above computes the Hecke representation $\psi(b) = \sum_{w \in \Sigma_n} a_w T_w$ in $O(n!(n \log n) \times N)$ operations and $O(n! \times N)$ algebraic operations in $\mathbb{Z}[q^{\pm 1}]$, storing at most $n! + O(1)$ Laurent polynomials in memory over the computation.*

Proof. The initialization of the vector `hecke` requires $O(n!)$ operations, and parsing the first crossing requires $O(n \log n)$ operations to convert permutations into their indices, and $O(1)$ arithmetic operations in $\mathbb{Z}[q^{\pm 1}]$ to initialize a constant number of entries in `hecke`. The induction step reads the array `hecke` (of length $n!$) once, and every time converts a constant number of permutations into their indices ($O(n \log n)$) and performs at most $O(1)$ operations in $\mathbb{Z}[q^{\pm 1}]$. Finally, we run the induction exactly n times. ◀

▶ Remark 11. Note that the algorithm above is rather simple, and the constant hidden by the big-O notation is rather small. This matters for running times in practice ; see Section 4.3.

▶ Remark 12. Note that the main **for** loop of Algorithm 1 can be naturally parallelized, considering that pairs of indices in `hecke`, corresponding to permutations w and $s_{i_j} w$, where s_{i_j} is the transposition corresponding to the new braid generator $\sigma_{i_j}^{\epsilon_j}$, are treated independently from other entries. We use a parallel and a non-parallel implementation of the **for** loop in Section 4.3.

4 Computing the HOMFLY-PT polynomial

4.1 The HOMFLY-PT polynomial

The HOMFLY-PT polynomial [15, 33] is a famous 2-variable knot invariant, that can be specialized to both the Jones polynomial and the Alexander polynomial. If one starts from a knot presented as a braid closure $\widehat{b}$ (see Figure 1), then, up to renormalization, the invariant is computed by taking a trace of the image of the braid b in the Hecke algebra. We now make this more precise, and we refer to the long version [29] for another presentation of the invariant.

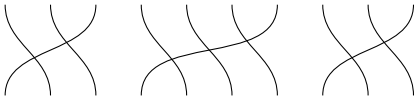


Figure 2 An annularly reduced element in Σ_{10}.

As in Section 3, we start from a braid $b = \sigma_{i_c}^{\epsilon_c} \cdots \sigma_{i_1}^{\epsilon_1}$, and pre-compute the array `hecke` encoding the Hecke representation $\psi(b) = \sum_{w \in \Sigma_n} a_w T_w$ with the algorithm of Theorem 10.

Our strategy is to simplify, in a single pass through `hecke`, the entries of the array, using the defining property of the trace, *i.e.*, $\mathrm{Tr}(\psi(bb')) = \mathrm{Tr}(\psi(b'b))$ for b, b' braids in B_n. We do not simplify the array all the way down, but instead only keep entries corresponding to permutations w with a simple form (called *annularly reduced*), and for which we know how to compute the participation to the final HOMFLY-PT polynomial.

Before describing the algorithm, we introduce some definitions and lemmas.

▶ **Definition 13.** *An element $w \in \Sigma_n$ is said to be* annularly reduced *(see Figure 2) if it is a product of disjoint cycles r_i so that:*

- *r_i has support $(j_i, j_i + 1, \cdots, j_i + l_i)$ for some j_i and l_i;*
- *$r_i = (j_i + l_i, j_i, j_i + 1, \cdots, j_i + l_i - 1)$.*

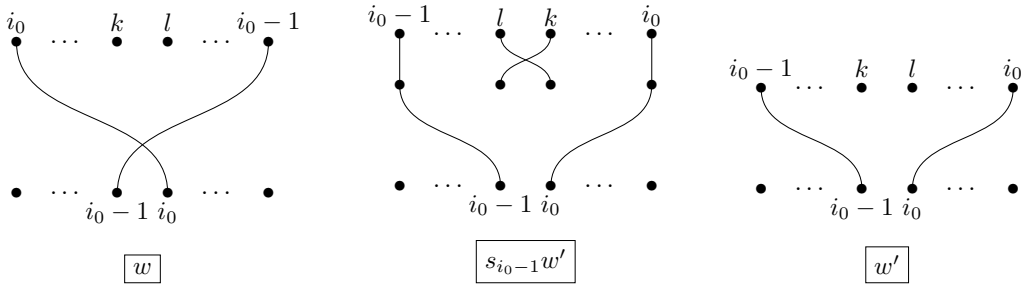
Our algorithm reduces braids to annularly reduced ones, using the following lemma.

▶ **Lemma 14.** *Consider $w \in \Sigma_n$, and define, if it exists:*

$$i_0 := \min\{i \mid w(i) < w(i) - 1\}.$$

Then $s_{i_0-1} w s_{i_0-1}$ and $w s_{i_0-1}$ are lower than w in the lexicographic order.

Proof. We first argue that $w = w' s_{i_0-1}$ as a reduced word. Indeed, from the definition of i_0, $w(i_0 - 1) \geq i_0 - 2$ and thus $w(i_0 - 1) \geq w_{i_0}$. The relative order of $i_0 - 1$ and i_0 is exchanged under w and thus $w = w' s_{i_0-1}$ with $\ell(w) = \ell(w') + 1$. We are thus in the following situation:

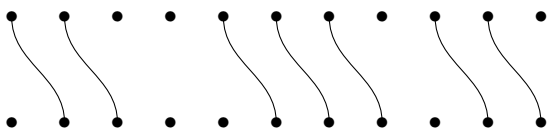


In both cases the lexicographic order gets lowered. ◀

The second piece of argument we need is the following one.

▶ **Lemma 15.** *If $w \in \Sigma_n$ is so that $w(i) \geq i - 1$ for all i, then w is annularly reduced.*

Proof. One first considers only the strands that move to the left. By hypothesis, those can only move one step to the left:



We are left with relating free ends with strands that can only go straight up or right. There is a unique solution, that consists of pairing the leftmost free end at the bottom with the leftmost free end at the top, and so on, which yields an annularly reduced element. ◀

We refer to the long version [29] for a precise definition of the HOMFLY-PT polynomial P. For our purpose, it suffices to say that the polynomial $P(\widehat{b})$ can be obtained from the vector `hecke` by:

- reducing the vector `hecke` in $\mathrm{Tr}(\mathcal{H}_n)$ using trace moves $b_1 b_2 = b_2 b_1$. This leaves us with $\sum_{w\in\mathcal{A}} a'_w T_w$ where $\mathcal{A}$ denotes the set of annularly reduced words in Σ_n from Definition 13;
- evaluating annularly reduced elements on $\prod_i (a^{1-l_i} q^{l_i-1} \frac{a-a^{-1}}{q-q^{-1}})$ (with the notations of Definition 13). See [29] for a justification.

4.1.1 Simplifying the trace

We now want to simplify the vector `hecke` in a trace process. Given a basis element T_w, we want, when possible, to simplify its image $\mathrm{Tr}(T_w) \in \mathrm{Tr}(\mathcal{H}_n)$. We do that by ordering elements in w by the lexicographic order $\leq_{\mathrm{lex}}$ of their one-line word $w(1)\cdots w(n)$, so that the simplification can be achieved in one scan through the set $\{T_w\}_{w\in\Sigma_n}$.

▶ Remark 16. Two elements that are annularly reduced might still be conjugate: the order of the cycles can be permuted. This results in keeping more non-zero entries in the vector we consider than necessary, but has the advantage of yielding a particularly simple algorithm.

We traverse the array `hecke` from right to left, *i.e.*, considering permutations w with *decreasing* position in lexicographic order: for every such w,

1. look for the first i_0 so that $w(i_0) < i_0 - 1$.
2. If there is no such i_0 then w is already annularly reduced (Lemma 15).
3. Otherwise it can be written $w = w' s_{i_0-1}$. Replace T_w by $T_{i_0-1} T'_w$ which expresses as a single or a sum of two basis elements (Lemma 7). In both cases, the elements that appear are strictly lower in the lexicographic order (Lemma 14), and the coefficient of $T_{i_0-1} T'_w$ is removed and added to entries of `hecke` to the left of w.

4.1.2 Evaluating the polynomial

We now are left with $v = \sum_{w\in\mathcal{A}} a'_w T_w$. The result we are looking for is:

$$P(\widehat{b}) = \sum_{w\in\mathcal{A}} a'_w P(\widehat{b_w}) \text{ (b_w is the positive lift of } w \in B_n).$$

The value for $P(\widehat{b_w})$ above is recorded in the following lemma.

▶ **Lemma 17.** *The HOMFLY-PT polynomial of an annularly reduced braid with cycles of lengths $l_1, \cdots, l_k$ is $\prod_{i=1}^{k} (a^{1-l_i} q^{l_i-1} \frac{a-a^{-1}}{q-q^{-1}})$.*

Proof. The computation consists of taking the product of each cycle value, as computed in the long version [29]. ◀

The last step of the algorithm thus consists in replacing T_w by the product of the values of the HOMFLY-PT polynomial on the cycles of T_w, and taking the sum for all w's, in a single pass through the array `hecke`.

4.2 Complexity and correctness

The correctness of the algorithm is guaranteed by the diverse lemmas proved along the way.

▶ **Theorem 18.** *Let* `hecke` *be the vector of (Laurent polynomials) coordinates of the Hecke representation of a braid $b \in B_n$, expressed in the $\{T_w\}_w$ basis. Given* `hecke`*, evaluating the HOMFLY-PT polynomial of b takes $O(n!(n \log n))$ operations and $O(n! \cdot n)$ algebraic operations, storing at most $n! + O(1)$ algebraic elements over the computation.*

Proof. Recall from Section 4.1.1 that we reduce the vector to only keep annularly reduced elements in a single pass on the array `hecke` of length $n!$. Computing w from its index and finding the first i_0 so that $w(i_0) < i_0 - 1$ takes $O(n \log n)$ operations. Then one finds w' so that $w = w' s_{i_0-1}$. This amounts to computing $w s_{i_0-1}$, which costs $O(1)$ operations. Then computing $T_{i_0-1} T_{w'}$ costs two multiplications in $\mathbb{Z}[q^{\pm 1}]$. In total, for this phase, we use $O(n! n \log n)$ combinatorial operations and $O(n!)$ arithmetic operations in $\mathbb{Z}[q^{\pm 1}]$.

Finally, thanks to Lemma 17, if suffices to know the number of cycles and their lengths to evaluate an annularly reduced element in $O(n)$ arithmetic operations in $\mathbb{Z}[q^{\pm 1}]$ at most (there are at most n disjoint cycles). Identifying cycles and their lengths is done directly by reading the one-line word of a permutation w ($O(n \log n)$ operations to compute the one-line word from w's index, then $O(n)$ to identify cycles), yielding a last step in $O(n! \cdot n \log n)$ combinatorial operations and $O(n! \cdot n)$ arithmetic operations in $\mathbb{Z}[q^{\pm 1}]$. Summing the complexity gives the theorem. ◀

▶ **Theorem 19.** *Given a braid $b \in B_n$ with N crossings, the above algorithm computes its HOMFLY-PT polynomial in $O(n!(n \log n) \times N)$ operations and $O(n! \times N)$ algebraic operations, storing $n! + O(1)$ algebraic elements.*

Proof. Summing the complexity of Theorem 10 and Theorem 18, all steps of the algorithm are covered; assuming $n \leq N$, Theorem 10 dominates the computation. ◀

4.3 Experimental performance

We have compared our code with the advanced `C++` library `Regina` [9], implementing Burton's state of the art algorithm [7] for the HOMFLY-PT polynomial. We have run our experiments on an 8 cores, 32Gb RAM computer; parallel implementations use the library `Intel::TBB`. We restrict our attention to braids whose closures lead to knots. The computation of the Hecke representation, which largely dominates the timings for our computation of the HOMFLY-PT polynomial, can be implemented in parallel, as described in Remark 12, while `Regina` does not admit a parallel implementation and the algorithm it implements may not parallelize as naturally as ours.

We have generated p-strands torus braids $T_{p,q}$ and p-strands weaving braids $W_{p,q}$, in their standard form (see Figure 3), keeping only those parameters q for which the closure is a knot. We have used them as is in our algorithm. `Regina` working with knot diagrams and treewidth, we have run simplification heuristics and the construction of the tree decomposition, before computing the HOMFLY-PT polynomial (and before starting the timer). However, for these knots, the braid presentation is expected to be minimal for both size and parameter.

In Figure 4 we compare the running time of our algorithm *not using parallelism*, against the algorithm from `Regina` for torus braids $T_{p,q}$ and weaving braids $W_{p,q}$, for $3 \leq p < 6$ and all values of $q = p + 1, \ldots, q_{\max}$, such that the closures of the braids lead to knots (*i.e.*, $\gcd(p, q) = 1$), with $q_{\max}$ picked such that the braids $T_{p,q_{\max}}$ and $W_{p,q_{\max}}$ have approximately

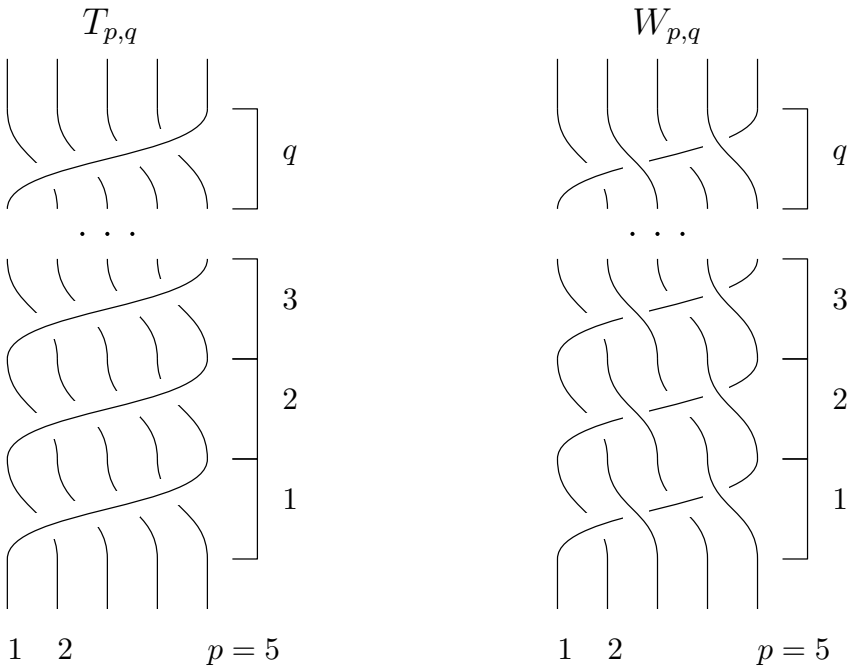


Figure 3 Torus braid $T_{p,q}$ and weaving braid $W_{p,q}$.

600 crossings. The results are very similar for torus and weaving braids, highlighting the fact that the running times depends mostly on the combinatorics of the input diagrams, and not so much on the algebraic computations. Our implementation consistently outperforms `Regina` on braids with more than 4 strands, by a factor ranging from 3 (shorter braids) to 9 (longer braids) for braids on 4 strands, a factor ranging from 3 to 4 for 5-braids.

In Figure 4, `Regina` is faster on the smaller 3-strands examples, by a factor 3 on shorter braids, and a smaller factor 1.3 on the larger braids ; all these timings are below 0.3 seconds, and may be explained by a slower initialization of data on our side. The global tendency is that our algorithm, even on its non-parallel form, gets better on longer braids (the slopes α of interpolating lines is $0.78 \leq \alpha \leq 0.8$ on all examples) and on braids with more strands.

On braids with ≥ 6 strands, `Regina` starts to `swap` due to excessive memory consumption, leading to impractical running times ; to the contrary, our implementation continues to be practical, Table 1 and see Figure 5 for much larger examples.

In Figure 5, we compare the performance of our implementation with and without parallelism. Recall that the parallelization is on the array `hecke` of length $n!$ (working with an n-strands braid). In consequence, we see the gain of parallelism increasing with the number of strands: first, the gain range from a factor 1.6 to 2.4 on 3-strands, from 3.3 to 4.54 4-strands, and from 3.8 to 6.25 on 5-strands. On braids with $6, 7, 8$-strands, the gain ranges between a factor 5 to 7.14, near the optimal 8 (on 8 cores); see Figure 5.

Finally, the memory consumption of our implementation is much more frugal than `Regina`, and remains very applicable for large number of strands. The memory cost depends mostly on the number of strands; however, for a fixed number of strands, longer braids tend to consume slightly more memory because the Laurent polynomials stored in the array `hecke` become more complicated; the memory consumption of longer braids is shown in Table 1).

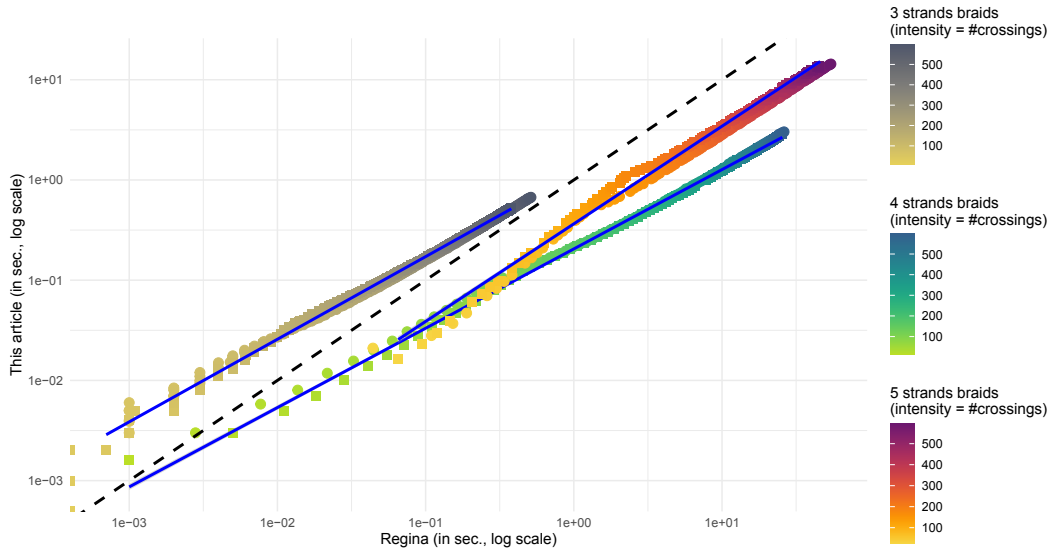


Figure 4 Timings of our algorithm (without parallelism) and `Regina`. Circle points represent torus braids, and square points represent weaving braids. The dashed line is $x = y$.

Table 1 Timing and peak memory usage of our algorithm using parallelism on large torus braids (similar results are found for weaving braids).

Braid	Num. of crossings	Timing parallel (8 cores)	Timing non-parallel	Peak memory
$T_{6,41}$	205	1.6 sec.	9.6 sec.	8.1 Mb
$T_{6,119}$	595	13.4 sec.	95.1 sec.	23.5 Mb
$T_{7,36}$	216	10.8 sec.	74 sec.	45.6 Mb
$T_{7,99}$	594	88.9 sec.	643 sec.	109 Mb
$T_{8,29}$	203	74.5 sec.	502 sec.	257 Mb
$T_{8,49}$	343	272 sec.	2265 sec.	435 Mb

In conclusion, our implementation is very competitive for knots represented by braids, even with a fairly large number of strands. It also combines very well with parallelism. We apply our parallel implementation in experimental mathematics in Section 5 to investigate algebraic properties of the Hecke algebra.

5 Proving the non-faithfulness of the Hecke representation in $\mathbb{Z}/m\mathbb{Z}$

With a bucket search procedure, coupled with the fast algorithm described in Section 3, we have been looking for non-trivial braids with trivial Hecke representation in several contexts.

Following [3, 18, 2], we generate a sample of non-trivial braids from B_n in Garside normal form, and store them in *buckets* of bounded size, one for each represented value of *augmented projlength* (see below). At iteration k of the search, the buckets store braids of Garside length k. For initialization, the Garside length 1 braids are in bijection with permutations in $\Sigma_n \setminus \{\mathrm{id}, \Delta\}$, and we store them in the buckets. Inductively, if we have generated a sample of Garside length k braids, we attempt to extend each of them into Garside length $k+1$ braids by appending an extra compatible Garside letter. Note that, when attempting to insert a new generated braid, we use a *reservoir sampling* [37] approach to guarantee that we extract an iid sample. In Section 5.3, we describe details on the optimized extension of

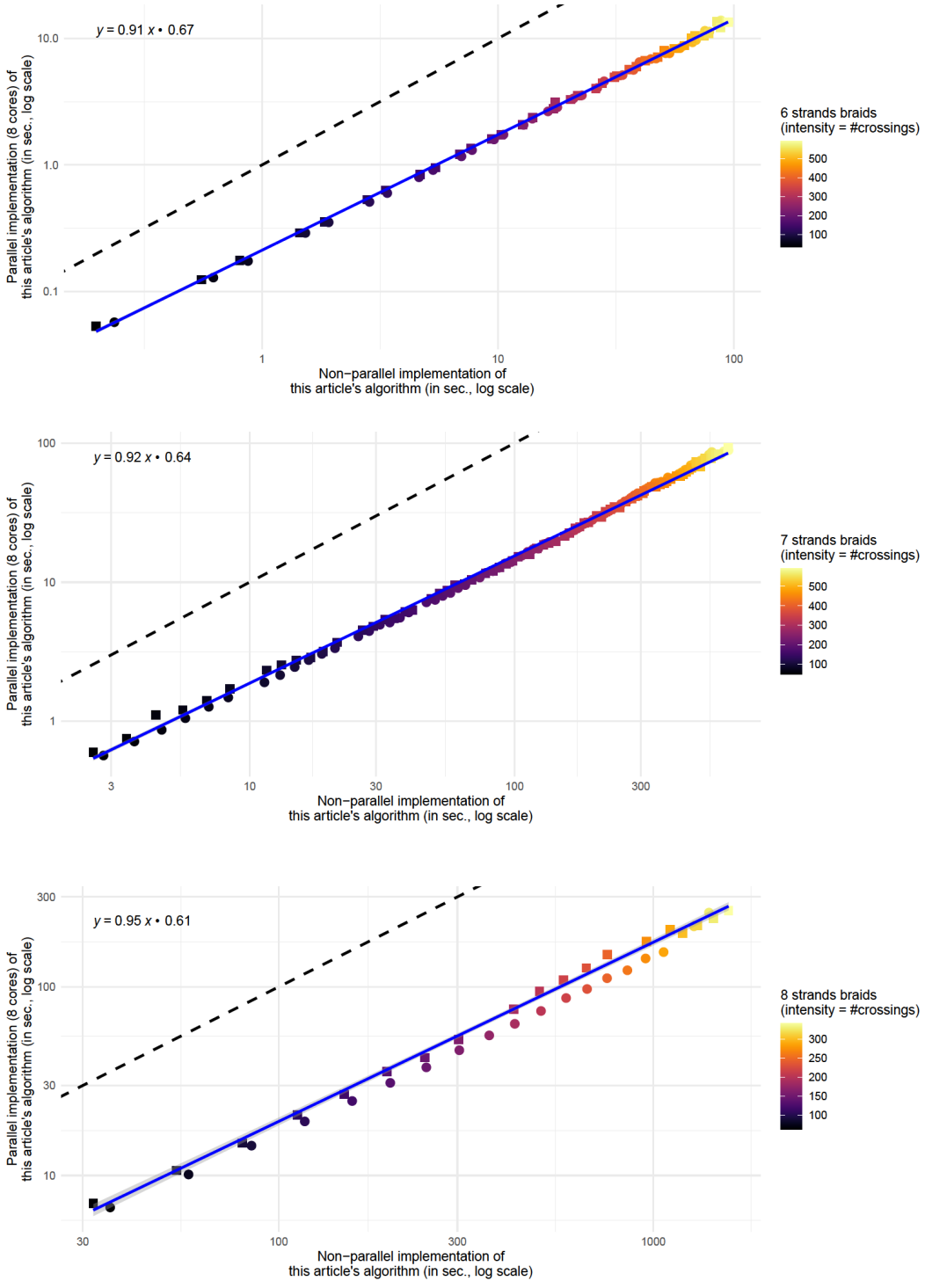


Figure 5 Timings of our algorithm with parallelism against our algorithm without parallelism. The interpolation line are near $x = y/7.14$ (we use 8 cores).

Garside normal forms we have implemented. In particular, we prove,

▶ **Lemma 20.** *Given a braid $b = b_k \dots b_1 \in B_n$ in Garside normal form, there is an $O(n \log n)$ algorithm to pick a Garside letter b_{k+1}* iid *among all Garside letters compatible with b_k.*

We define the *augmented projlength* of a positive braid b to be

$$\widehat{\text{projlength}}(b) : \sup_{k \leq 0}\{\text{projlength}(\psi(\Delta^k b))\},$$

and we try to minimize this quantity over the search. To avoid computing the projlength for infinitely many negative powers of Δ, we (heuristically) stop the computation of $\widehat{\text{projlength}}(b)$ when we encounter a power $k_0 < 0$ for which $\psi(\Delta^k b)$ has a coordinate with negative degree.

The bucket search algorithm, if successful, finds positive braids b (not of the form Δ^l) such that there exists of negative power k of Δ satisfying: $\Delta^k \cdot b$ has projlength $= 0$. In all cases where we found such braids, they had an image in the Hecke algebra (in the basis used from Lemma 6) with a single non-trivial coordinate idx equal to 1. This coordinate corresponding to the positive permutation s^{-1}, the braid $s^{-1} \cdot \Delta^k \cdot b$ has trivial Hecke representation.

We have run our searches in parallel on a cluster of 2 CPUs, 128 cores/CPU, and 1024GB RAM.

5.1 Non-faithful Hecke representations of B_4

The decomposition of $\mathcal{H}_4$ into irreducible B_4 representations only contains the Burau representation as possibly faithful summand. The latter being not faithful at $p = 2, 3, 5$ implies the same for the Hecke representation. We indeed found such counterexamples for $p = 2$ and 3, as well as with $p = 4$, that we present in the long version [29].

5.2 Non-faithful Hecke representations of B_5

The same search performed on B_5 with base ring $\mathbb{Z}/2\mathbb{Z}$ also yielded a counterexample to faithfulness. In this case, non-faithfulness follows by including the B_4 kernel into B_5. Here we could run a direct search in B_5 to find *explicit* counterexamples.

▶ **Theorem 21.** *The braid $s_9^{-1}\Delta^{-9}b$ is in the kernel of the Hecke representation of B_5 with coefficients in $\mathbb{Z}/2\mathbb{Z}[q^{\pm 1}]$, with:*

$$\begin{aligned} b =& \sigma_3\sigma_4\sigma_2\sigma_3\sigma_1\sigma_2\sigma_2\sigma_4\sigma_2\sigma_3\sigma_1\sigma_2\sigma_4\sigma_2\sigma_3\sigma_3\sigma_2\sigma_3\sigma_1\sigma_3\sigma_1\sigma_2\sigma_2\sigma_3\sigma_1\sigma_3\sigma_1\sigma_2\sigma_2\sigma_3\sigma_1\sigma_3\sigma_1\sigma_2\sigma_4\sigma_3 \\ & \sigma_2\sigma_3\sigma_4\sigma_1\sigma_4\sigma_3\sigma_2\sigma_1\sigma_2\sigma_3\sigma_4\sigma_4\sigma_3\sigma_2\sigma_3\sigma_1\sigma_2\sigma_3\sigma_3\sigma_2\sigma_3\sigma_4\sigma_1\sigma_2\sigma_3\sigma_4\sigma_4\sigma_3\sigma_4\sigma_2\sigma_1\sigma_2\sigma_3\sigma_3\sigma_4\sigma_2 \\ & \sigma_1\sigma_2\sigma_3\sigma_4\sigma_4\sigma_3\sigma_4\sigma_1\sigma_2\sigma_3\sigma_3\sigma_4\sigma_2\sigma_3\sigma_4\sigma_1\sigma_4\sigma_3\sigma_4\sigma_1\sigma_2 \\ s_9^{-1} =& \sigma_2^{-1}\sigma_3^{-1}\sigma_4^{-1}, \qquad \Delta^{-1} = \sigma_4^{-1}\sigma_3^{-1}\sigma_2^{-1}\sigma_1^{-1}\sigma_4^{-1}\sigma_3^{-1}\sigma_2^{-1}\sigma_4^{-1}\sigma_3^{-1}\sigma_4^{-1}. \end{aligned}$$

b is of Garside length 19*, and the resulting counterexample is a braid with* 186 *crossings.*

5.3 Garside automaton

To generate braids, we use the following finite state automaton.

▶ **Definition 22.** *Let* **Garside** *be the finite state automaton with:*

- *states, permutations s in Σ_n (or equivalently elements in $[1, \Delta]$);*
- *s'-labeled arrows $s \to s'$ if $\forall s_j \leq_R s'$ then $s_j \leq_L s$.*

We create increasingly complicated braids by applying to a Garside normal form $b_k \cdots b_1$ any (possibly randomly chosen) letter from $\mathbf{Garside}(b_k)$. The construction of the automaton can be made very efficient thanks to the following observation.

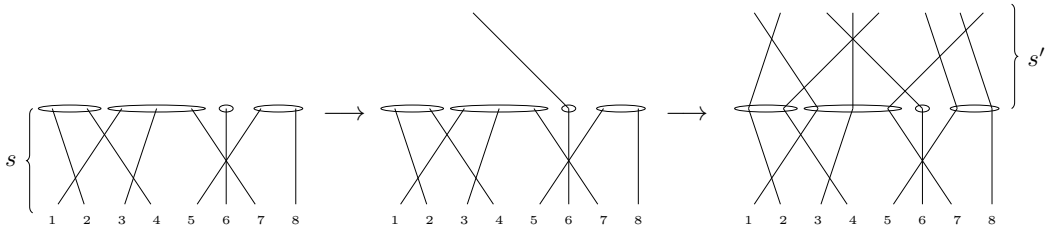
▶ **Lemma 23.** *Given $s = (s(1), \cdots, s(n))$ and $s' = (s'(1), \cdots, s'(n))$, s' can be applied at the state s if and only if the following property holds:*

$s' \neq \mathrm{id}$ *and* $\forall i \in [\![1, n]\!]$ *with* $s'(i) > s'(i+1),\ s^{-1}(i) > s^{-1}(i+1)$.

Proof. This restates the fact that right descents from s' need to be left descents of s. ◀

From a state s, we want to generate admissible arrows s'. All pairs $\{(i_r, i_r + 1) \mid \exists l < k,\ i_r = s(k),\ i_r + 1 = s(l)\}$ partition $[\![1, n]\!]$ into intervals $[\![i_r + 1, i_{r+1}]\!]$. Lemma 23 prevents two entries from the same interval to cross in s'. This suggests the following (see Figure 6):

- for length 1 intervals ($i_r + 1 = i_{r+1}$), freely choose $s'(i_r)$ in $[\![1, n]\!]$ (forming S_{free});
- images of the other intervals are disjoint ordered subsets of $[\![1, n]\!] \setminus S_{free}$. These choices can be made one interval at a time in any order. Starting from the first interval $[\![i_r + 1, i_{r+1}]\!]$ of length $p' > 1$, one chooses a random ordered subset $S_{p'}$ of p' elements in $[\![1, n - p]\!]$, and then decide that $s'(i_r + k)$ is the $S_{p'}(k)$-th element in $[\![1, n]\!] \setminus S_{free}$. Then one replaces S_{free} by $S_{free} \cup S_{p'}$ and goes to the next interval of size greater than 1.



Start from s. Choose S_{free}. Choose the other images so that no strands coming out of the same ellipse cross.

Figure 6 Generating descents.

A major advantage of the above construction is the following.

▶ **Lemma 24.** *Given a braid $b \in B_n$ in Garside normal form $b_k \dots b_1$, there is an $O(n \log n)$ algorithm to pick a Garside letter b_{k+1} iid among all Garside letters compatible with b_k.*

Proof. If one can make uniform random choices of p unordered elements in $[\![1, n]\!]$, and uniform random choices of p ordered elements in $[\![1, p']\!]$, then the procedure described above the lemma produces uniform choices in $\mathbf{Garside}(s)$. Picking iid random subsets can be done in time $O(n \log n)$ using, for example, a reservoir sampling algorithm. ◀

References

1 J. W. Alexander. A lemma on systems of knotted curves. *Proceedings of the National Academy of Sciences*, 9(3):93–95, March 1923. doi:10.1073/pnas.9.3.93.

2 A. Bapat and H. Queffelec. Some remarks about the faithfulness of the Burau representation of Artin–Tits groups, 2024. arXiv:2409.00144.

3 S. Bigelow. The Burau representation is not faithful for n= 5. *Geometry & Topology*, 3(1):397–404, 1999.

4 S. Bigelow. Braid groups and iwahori-hecke algebras. *Problems on mapping class groups and related topics*, 74:285–299, 2006. arXiv:1411.5418.

5 M. Brittenham and S. Hermiller. Unknotting number is not additive under connected sum, 2025. arXiv:2506.24088.

6 B A Burton. Introducing Regina, the 3-manifold topology software. *Experiment. Math.*, 2004.

7 Benjamin A. Burton. The HOMFLY-PT Polynomial is Fixed-Parameter Tractable. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2018.18.

8 Benjamin A. Burton. The Next 350 Million Knots. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2020.25.

9 Benjamin A. Burton, Ryan Budney, William Pettersson, et al. Regina: Software for 3-manifold topology and normal surface theory. http://regina.sourceforge.net/, 1999–2018.

10 Benjamin A. Burton, Clément Maria, and Jonathan Spreer. Algorithms and complexity for Turaev–Viro invariants. *Journal of Applied and Computational Topology*, 2(1–2):33–53, August 2018. doi:10.1007/s41468-018-0016-2.

11 Benjamin A. Burton and Melih Ozlen. A fast branching algorithm for unknot recognition with experimental polynomial-time behaviour, 2014. arXiv:1211.1079.

12 M Culler, N M Dunfield, and J R Weeks. SnapPy, a computer program for studying the geometry and topology of 3-manifolds. http://snappy.computop.org/, 1991–2018.

13 P. Dehornoy, F. Digne, E. Godelle, D. Krammer, and J. Michel. *Foundations of Garside theory*, volume 22. European Mathematical Society Zürich, 2015.

14 Renaud Detcherry. A quantum obstruction for purely cosmetic surgeries. *Annales de l'Institut Fourier*, 2025. Online first. doi:10.5802/aif.3673.

15 P. Freyd, D. Yetter, J. Hoste, W. B. R. Lickorish, K. Millett, and A. Ocneanu. A new polynomial invariant of knots and links. *Bull. Am. Math. Soc., New Ser.*, 12:239–246, 1985. doi:10.1090/S0273-0979-1985-15361-3.

16 Stavros Garoufalidis and Yueheng Lan. Experimental evidence for the volume conjecture for the simplest hyperbolic non-2–bridge knot. *Algebraic & Geometric Topology*, 5(1):379–403, May 2005. doi:10.2140/agt.2005.5.379.

17 F. A Garside. The braid group and other groups. *The Quarterly Journal of Mathematics*, 20(1):235–254, 1969.

18 J. Gibson, G. Williamson, and O. Yacobi. 4-Strand Burau is Unfaithful Modulo 5, 2023. arXiv:2310.02403v1.

19 Joel Hass, Jeffrey C. Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *Journal of the ACM*, 46(2):185–211, March 1999. doi:10.1145/301970.301971.

20 T. Ito. A kernel of a braid group representation yields a knot with trivial knot polynomials. *Mathematische Zeitschrift*, 280(1):347–353, 2015. arXiv:1402.2028.

21 V. FR Jones. Hecke algebra representations of braid groups and link polynomials. *Annals of Math.*, 126:335–388, 1987.

22 Louis H. Kauffman. State models for link polynomials. *Enseignement Mathématique*, 36(1–2):1–37, 1990.

23 Greg Kuperberg. How hard is it to approximate the Jones polynomial? *Theory of Computing*, 11(1):183–219, 2015. doi:10.4086/toc.2015.v011a006.

24 Marc Lackenby. The efficient certification of knottedness and thurston norm. *Advances in Mathematics*, 387:107796, August 2021. doi:10.1016/j.aim.2021.107796.

25 G. I Lehrer and N. Xi. On the injectivity of the braid group in the hecke algebra. *Bulletin of the Australian Mathematical Society*, 64(3):487–493, 2001.

26 Corentin Lunel and Arnaud de Mesmay. A Structural Approach to Tree Decompositions of Knots and Spatial Graphs. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:16, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2023.50.

27 J.A. Makowsky and J.P. Mariño. The parametrized complexity of knot polynomials. *Journal of Computer and System Sciences*, 67(4):742–756, 2003. Parameterized Computation and Complexity 2003. doi:10.1016/S0022-0000(03)00080-1.

28 Clément Maria. Parameterized Complexity of Quantum Knot Invariants. In Kevin Buchin and Éric Colin de Verdière, editors, *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 53:1–53:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2021.53.

29 Clément Maria and Hoel Queffelec. A fast algorithm for the hecke representation of the braid group, and applications to the computation of the homfly-pt polynomial and the search for interesting braids, 2025. arXiv:2512.06142.

30 A. Mathas. *Iwahori-Hecke algebras and Schur algebras of the symmetric group*, volume 15 of *Univ. Lect. Ser.* Providence, RI: American Mathematical Society, 1999.

31 H. Morton. br9z. https://www.liverpool.ac.uk/~su14/knotprogs.html, 1985.

32 Hugh R Morton and HB Short. Calculating the 2-variable polynomial for knots presented as closed braids. *Journal of Algorithms*, 11(1):117–131, 1990. doi:10.1016/0196-6774(90)90033-B.

33 J. H. Przytycki and P. Traczyk. Invariants of links of Conway type. *Kobe J. Math.*, 4(2):115–139, 1987.

34 Saul Schleimer, Arnaud de Mesmay, Jessica S. Purcell, and Eric Sedgwick. On the tree-width of knot diagrams. *J. Comput. Geom.*, 10(1):164–180, 2019. doi:10.20382/JOCG.V10I1A6.

35 C. C Squier. Matrix representations of artin groups. *Proceedings of the American Mathematical Society*, 103(1):49–53, 1988.

36 The KumQuAT Project. KumQuAT: Algorithms in quantum topology, 2026. URL: https://kumquat.inria.fr/.

37 Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985. doi:10.1145/3147.3165.

Robust Algorithms for Path and Cycle Problems in Geometric Intersection Graphs

Malory Marin
ENS de Lyon, CNRS, Université Claude Bernard Lyon 1,
LIP, UMR 5668, 69342, Lyon cedex 07, France

Jean-Florent Raymond
CNRS, ENS de Lyon, Université Claude Bernard Lyon 1,
LIP, UMR 5668, 69342, Lyon cedex 07, France

Rémi Watrigant
Université Claude Bernard Lyon 1, CNRS, ENS de Lyon,
LIP, UMR 5668, 69342, Lyon cedex 07, France

Abstract

We study the design of robust subexponential algorithms for classical connectivity problems on intersection graphs of similarly sized fat objects in $\mathbb{R}^d$. In this setting, each vertex corresponds to a geometric object, and two vertices are adjacent if and only if their objects intersect. We introduce a new tool for designing such algorithms, which we call a *λ-linked partition*. This is a partition of the vertex set into groups of highly connected vertices. Crucially, such a partition can be computed in polynomial time and does not require access to the geometric representation of the graph.

We apply this framework to problems related to paths and cycles in graphs. First, we obtain the first *robust ETH-tight* algorithms for Hamiltonian Path and Hamiltonian Cycle, running in time $2^{O(n^{1-1/d})}$ on intersection graphs of similarly sized fat objects in $\mathbb{R}^d$. This resolves an open problem of de Berg *et al.* [STOC 2018] and completes the study of these problems on geometric intersection graphs from the viewpoint of ETH-tight exact algorithms.

We further extend our approach to the parameterized setting and design the first robust subexponential parameterized algorithm for Long Path in any fixed dimension d. More precisely, we obtain a randomized robust algorithm running in time $2^{O(k^{1-1/d}\log^2 k)}\, n^{O(1)}$ on intersection graphs of similarly sized fat objects in $\mathbb{R}^d$, where k is the natural parameter. Besides λ-linked partitions, our algorithm also relies on a low-treewidth pattern covering theorem that we establish for geometric intersection graphs, which may be viewed as a refinement of a result of Marx-Pilipczuk [ESA 2017]. This structural result may be of independent interest.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Robust algorithms, geometric intersection graphs, subexponential FPT algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.77

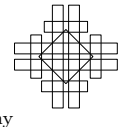
Related Version *Full Version*: https://arxiv.org/pdf/2512.03843

Funding *Malory Marin*: Supported by Projet ANR GODASse, Projet-ANR-24-CE48-4377.

1 Introduction

Given a set $\mathcal{F}$ of objects in $\mathbb{R}^d$, the *intersection graph* of $\mathcal{F}$ is defined as the graph having one vertex for each object in $\mathcal{F}$, and an edge between two vertices whenever the corresponding objects intersect. One of the most extensively studied classes of intersection graphs are the *unit disk graphs*, obtained when the objects are disks in $\mathbb{R}^2$ of identical radius. In this work, we consider families $\mathcal{F}$ of *similarly sized β-fat* objects in $\mathbb{R}^d$ (for some constants $d, \beta \geq 1$) which means that for every object $O \in \mathcal{F}$, there are two balls B_{in} and B_{out} of $\mathbb{R}^d$ such that $B_{\text{in}} \subseteq O \subseteq B_{\text{out}}$, where B_{in} has radius 1 and B_{out} has radius β.


© Malory Marin, Jean-Florent Raymond, and Rémi Watrigant;
licensed under Creative Commons License CC-BY 4.0




In their seminal work, de Berg *et al.* [4] introduced a general framework for deriving algorithms with running times that are tight under the Exponential Time Hypothesis (ETH) for a broad range of classical problems on intersection graphs of similarly sized fat objects. Their framework encompasses problems such as Maximum Independent Set, Dominating Set, Steiner Tree, and Hamiltonian Cycle. The key idea of their approach is the construction of a partition $\mathcal{P} = (V_1, \ldots, V_t)$ of the vertex set of the input graph G satisfying the following properties:

- **(i)** each induced subgraph $G[V_i]$ can be further partitioned into at most κ cliques, and
- **(ii)** the quotient graph of the parts, denoted $G_{\mathcal{P}}$, has bounded degree and treewidth $O(n^{1-1/d})$.

Notice that $\mathcal{P}$ can be computed *without* access to the geometric representation of F, and is enough to solve all the aforementioned problems except for Hamiltonian Cycle. Algorithms of this type, which operate solely on the intersection graph and do not require a geometric representation, are referred to as *robust* algorithms. We stress here that robustness is a substantial advantage since for many classes of intersection graphs such as unit disk graphs, computing a representation is $\exists\mathbb{R}$-complete [10].

However, the algorithm of [4] solving Hamiltonian Cycle requires one additional step: computing a partition of each V_i into a bounded number of cliques. For this step, de Berg *et al.* used the geometric representation. They explicitly left open the question of whether a robust ETH-tight algorithm for Hamiltonian Cycle could be obtained. One possible direction toward such a result would be to design a robust algorithm that computes a partition into a constant number of cliques whenever the input graph is known to contain one. Unfortunately, recent advances on clique partitions of geometric graphs suggest that this task is likely to be difficult [11].[1] It is worth noting that, in the case of unit disk graphs, such a clique partition can be obtained using the approach of [13].

We circumvent this problem by using a new type of partition into highly connected subgraphs called λ-*linked partitions*. We show the two following crucial properties : these partitions can be used to solve the considered problems (instead of partitions into cliques) and they can be computed in polynomial time (and, importantly, without relying on the geometric representation) in graphs known to admit a partition into a constant number of cliques.

Combining these findings with the framework of de Berg *et al.*, we obtain the following.

▶ **Theorem 1.** *For every constants $d \geqslant 1$ and $\beta \geqslant 1$ there is a robust algorithm solving Hamiltonian Path (resp. Hamiltonian Cycle) in time $2^{O\left(n^{1-1/d}\right)}$ on intersection graphs of similarly sized β-fat objects in $\mathbb{R}^d$.*

This result matches the ETH-based lower bound of [4] and resolves an open problem from the same paper.

To further demonstrate the applicability of our techniques, we show in a second part of the paper how they can be used to deal with parameterized version of Hamiltonian Path, that is, Long Path parameterized by the length of the path. For this problem and for Long Cycle, Fomin *et al.* proved in [16] that there are algorithms running in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$ on intersection graphs of similarly sized fat objects in $\mathbb{R}^2$. Their approach relies on computing a *clique partition* of the vertex set and analyzing the resulting *clique-grid graph*. The idea of using clique partitions in the context of subexponential parameterized algorithms was

[1] There is no $2^{o(n)}$-time algorithm for this problem in unit ball graphs of $\mathbb{R}^5$, unless the ETH fails [11].

introduced in [8] and has since become a central tool in the area [15]. However, the clique-grid graph is typically defined using a geometric representation of the input graph. Designing a robust subexponential parameterized algorithm for Long Path in dimensions $d \geqslant 3$ has remained an open problem. Indeed, while the bidimensionality-based approach of [16] does not generalize to higher dimensions, the approach of [8], based on Baker's technique [5], again requires access to the geometric representation.

We show that this dependency can be removed by combining two key techniques. The first consists, as in the case of the Hamiltonian Path problem, in replacing the clique partition with a partition into highly connected parts. The second is an adaptation of the *low-treewidth pattern covering* technique, introduced by Fomin *et al.* [7] for planar graphs and later extended to graphs of polynomial growth by Marx and Pilipczuk [12]. Formally, we prove that there exists a randomized polynomial-time algorithm which, given an intersection graph of similarly sized objects in $\mathbb{R}^d$ with bounded degree and an integer k, outputs a vertex set $A \subseteq V(G)$ such that $G[A]$ has treewidth $O(k^{1-1/d} \log k)$, and for every set $X \subseteq V(G)$ of size at most k, we have $X \subseteq A$ with probability at least $2^{-\Omega(k^{1-1/d} \log^2 k)}$. This result can be viewed as a strengthening of that of Marx and Pilipczuk [12] in the case where the input graph has additional geometric structure rather than merely polynomial growth. Combined together, these techniques yield the first robust subexponential FPT algorithm for Long Path.

▶ **Theorem 2.** *For every constants $d \geqslant 1$ and $\beta \geqslant 1$, there is a robust randomized parameterized algorithm solving Long Path in time $2^{O(k^{1-1/d} \log^2 k)} n^{O(1)}$ on intersection graphs of similarly sized β-fat objects of $\mathbb{R}^d$.*

2 Preliminaries

2.1 Graph Theory

Unless otherwise specified we use standard graph theory terminology. All graphs considered in this paper are simple, undirected, and finite. For a graph G, we denote by $V(G)$ and $E(G)$ its vertex set and edge set, respectively. We write $n = |V(G)|$ and $m = |E(G)|$ its number of vertices and edges when G is clear from the context. For a vertex $v \in V(G)$, we write $N_G(v)$ for its *open neighborhood*, that is, the set of vertices adjacent to v, and $N_G[v] = N_G(v) \cup \{v\}$ for its *closed neighborhood*. For $X \subseteq V$, we denote by $G[X]$ the subgraph of G induced by X, and by $G - X$ the subgraph induced by $V \setminus X$. The *degree* of a vertex v is $d_G(v) = |N_G(v)|$, and the maximum degree of G is denoted by $\Delta(G)$. An *independent set* $I \subseteq V(G)$ is a subset of pairwise non-adjacent vertices of G. The *independence number* of G, denoted by $\alpha(G)$, is the maximum size of an independent set of G.

A *path* in a graph G is a sequence of distinct vertices $(v_1, v_2, \ldots, v_k)$ such that any two consecutive vertices are adjacent in G. A *cycle* is also a sequence of vertices $(v_1, v_2, \cdots v_k)$ such that any two consecutive vertices are adjacent in G, and $v_k v_1 \in E(G)$. Given a path P (resp. a cycle), we denote by $V(P)$ the set of vertices appearing in the sequence.

For two vertices u, v of a graph G, the *distance* between u and v, denoted by $\mathrm{dist}_G(u, v)$, is the minimum number of edges on a path from u to v in G. For an integer r, and a vertex $v \in V(G)$, we define $B_G(v, r) = \{ w \in V(G) : \mathrm{dist}_G(v, w) < r \}$, that is, the set of vertices at distance strictly less than r from v in G. Similarly, we let $\partial B_G(v, r) = \{ w \in V(G) : \mathrm{dist}_G(v, w) = r \}$ denote the set of vertices at distance exactly r from v.

2.2 Treewidth, κ-partition and $\mathcal{P}$-contraction

A *tree decomposition* of a graph $G = (V, E)$ is a pair $(T, \{X_t\}_{t \in V(T)})$ where T is a tree, $\{X_t\}_{t \in V(T)}$ is a collection of subsets of $V(G)$ (called *bags*), and satisfying the following:

(i) $\bigcup_{t \in V(T)} X_t = V(G)$,
(ii) for every edge $uv \in E(G)$, there exists a bag X_t containing both u and v, and
(iii) for every vertex $v \in V$, the set $\{t \in V(T) \mid v \in X_t\}$ induces a connected subtree of T.

Given a graph G and a weight function $\gamma : V(G) \to \mathbb{R}_{\geqslant 0}$, the *weighted width* of a tree decomposition $(T, \{X_t\})$ is $\max_{t \in V(T)} \sum_{v \in X_t} \gamma(v)$. The *weighted treewidth of* G (w.r.t. a given weight function) is the minimum weighted treewidth over all tree decompositions of G [14].

In [4], de Berg *et al.* introduced the notion of a κ-partition as a generalization of a clique partition, in order to obtain separator theorems for intersection graphs of similarly sized fat objects in $\mathbb{R}^d$.

▶ **Definition 3** (κ-partition [4]). *Let G be a graph and let $\kappa \geqslant 1$ be an integer. A* κ-partition *of G is a partition $\mathcal{P} = (V_1, \ldots, V_t)$ of $V(G)$ such that each class V_i induces a connected subgraph $G[V_i]$ whose vertex set can be partitioned into at most κ cliques. If $\kappa = 1$, $\mathcal{P}$ is called a* clique partition *of G.*

▶ **Definition 4** ($\mathcal{P}$-contraction [4]). *Given a graph G and a partition $\mathcal{P} = (V_1, \ldots, V_t)$ of $V(G)$, the* $\mathcal{P}$-contraction *of G, denoted by $G_{\mathcal{P}}$, is the (simple) graph whose vertex set is $\{V_1, \ldots, V_t\}$, where V_iV_j is an edge if and only if there is an edge in G between a vertex of V_i and a vertex of V_j. The* degree *of the partition $\mathcal{P}$ refers to the maximum degree of $G_{\mathcal{P}}$. Given a weight function $\gamma : \mathbb{R} \to \mathbb{R}$, we define the weighted treewidth of $G_{\mathcal{P}}$ (w.r.t. γ) by assigning to each vertex $V_i \in V(G_{\mathcal{P}})$ the weight $\gamma(|V_i|)$.*

The following result shows that for intersection graphs of similarly sized fat objects, it is possible to construct a κ-partition for which $G_{\mathcal{P}}$ has bounded maximum degree and sublinear weighted treewidth.

▶ **Theorem 5** (de Berg *et al.* [4, Theorem 12]). *Let $d \geqslant 2$, $\beta \geqslant 1$ and $\varepsilon > 0$ be constants, and let γ be a weight function satisfying $1 \leqslant \gamma(x) = O(x^{1-1/d-\varepsilon})$. There exist constants κ and Δ such that the following holds :*

- *Any intersection graph G of similarly-sized β-fat objects in $\mathbb{R}^d$ has a κ-partition $\mathcal{P}$ such that $G_{\mathcal{P}}$ has weighted treewidth $O(n^{1-1/d})$ (with respect to γ) and maximum degree Δ.*
- *There is a polynomial-time algorithm that, given such a graph G (without the representation), returns such a κ-partition, and a $2^{O(n^{1-1/d})}$-time algorithm which returns the corresponding tree decomposition.*

▶ Remark 6. We provide here a few additional observations regarding the construction and properties of the κ-partition.

- The κ-partition can be obtained in a *greedy manner* as follows: consider a maximal independent set S of G and create one part for each vertex of this independent set. By maximality, every vertex $v \in V(G) \setminus S$ has a neighbor in S. We pick any $u \in N(v) \cap S$ and add v to the same part as u. As a consequence, in any geometric representation of G with similarly sized β-fat objects, two objects belonging to the same part of a κ-partition can be assumed to be at distance at most 2β (see Figure 1).
- The graph $G_{\mathcal{P}}$ defined in the theorem above has (unweighted) treewidth $O(n^{1-1/d})$ when γ is the unit function.

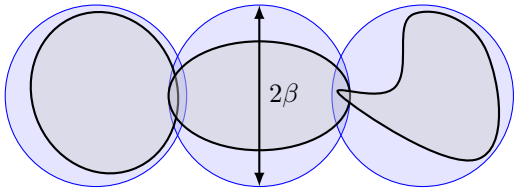


Figure 1 An example of 3 fat objects, the blue disks representing the enclosing disks of each object. The objects on left and right intersect the same object so there are at distance at most 2β.

2.3 Hamiltonicity and Connectivity

A path (or cycle) is said to be *Hamiltonian* if it contains all vertices of the graph. The Hamiltonian Path (resp. Cycle) problem asks, given a graph G, whether G has a Hamiltonian path (resp. cycle). The Long Path problem asks, given a graph G and a parameter k, whether G has a path on k vertices.

Since the problems we study are either trivially infeasible on disconnected graphs, or can be handled by considering each connected component independently, we restrict our attention to connected graphs throughout the paper.

A *vertex separator* in a connected graph G is a subset $A \subseteq V(G)$ such that $G - A = G[V(G) \setminus A]$ is disconnected. The *vertex connectivity* of G, denoted by $c_v(G)$, is the size of a minimum vertex separator of G, or $|V(G)| - 1$ if G is complete. Since we will not consider edge connectivity, we simply refer to $c_v(G)$ as the *connectivity* of G. A graph G is said ℓ-connected whenever $\ell \leqslant c_v(G)$.

A graph G is said to be *ℓ-linked* if $|V(G)| \geqslant 2\ell$ and, for every collection of ℓ disjoint pairs of distinct vertices $(s_1, t_1), (s_2, t_2), \ldots, (s_\ell, t_\ell)$, there exist ℓ vertex-disjoint paths $P_1, \ldots, P_\ell$ such that each P_i connects s_i to t_i. The graph G is *Hamiltonian-ℓ-linked* if there are such paths that, in addition, span all the vertices of G.

Let H and G be graphs. A *topological minor embedding* (or *TM-embedding*) of H into G is a pair (M, f), where M is a subgraph of G and $f : V(H) \to V(M)$ is an injective mapping such that, for every edge $\{u, v\} \in E(H)$, there exists a path P_{uv} in M connecting $f(u)$ and $f(v)$, and all these paths are pairwise internally vertex-disjoint. We say that H is a *topological minor* of G if such a TM-embedding exists. Moreover, when $V(G) = \bigcup_{uv \in E(H)} V(P_{u,v})$, we say that the TM-embedding is *spanning* all vertices of G.

The following lemma, due to Fomin *et al.* [6], establishes that in highly connected graphs, either a given small graph H can be TM-embedded in a spanning way, or there exists a large independent set.

▶ **Lemma 7** ([6, Lemma 4]). *Let H and G be graphs, with H non-empty. Let $f : V(H) \to V(G)$ be an injective mapping, and let k be a positive integer. Assume that G is $(\max\{k+2, 10\} \cdot h)$-connected, where $h = |V(H)| + |E(H)|$. Then there exists an algorithm with running time $2^{(h+k)^{O(k)}} + |G|^{O(1)}$ that computes either a subgraph $M \subseteq G$ such that (M, f) is a TM-embedding of H in G spanning all vertices of G, or an independent set of size k in G.*

Note that when H is the disjoint union of ℓ edges and $k > \alpha(G)$, Lemma 7 implies that the algorithm returns a TM-embedding of H spanning all vertices of G, regardless of the choice of the injective mapping f. It is straightforward to observe that in this case, H is Hamiltonian-ℓ-linked.

▶ **Lemma 8** (Consequence of Lemma 7). *Let G be a graph and let $\ell \geq 1$ be an integer. If G is $(\max\{\alpha(G) + 3, 10\} \cdot 3\ell)$-connected, then G is Hamiltonian-ℓ'-linked for all $\ell' \leqslant \ell$. Moreover, there exists an algorithm with running time $g(\alpha(G), \ell) \cdot |G|^{O(1)}$ (for some computable function*

g) which, given a graph G and 2ℓ distinct vertices $\{s_i, t_i\}_{1 \leqslant i \leqslant \ell}$, constructs ℓ vertex-disjoint paths $P_1, \ldots, P_\ell$ such that each P_i connects s_i to t_i, and the union of these paths spans all vertices of G.

3 A Robust Algorithm for Hamiltonian Path and Cycle

This section is devoted to our main tool, namely λ-linked partitions of intersection graphs of similarly sized fat objects, and to their application in proving Theorem 1.

3.1 Linked Partitions

We now generalize the notion of κ-partition to the concept of a *λ-linked partition*, where each part has to be highly connected.

▶ **Definition 9** (λ-linked partition). *Let G be a graph and let $\lambda \geqslant 1$ be an integer. A* λ-linked partition *of G is a partition $\mathcal{P} = (V_1, \ldots, V_t)$ of $V(G)$ such that each part V_i either induces a clique or a Hamiltonian λ-linked graph.*

▶ **Theorem 10.** *Let $d \geqslant 2$ and $\beta \geqslant 1$ be constants. There exists a constant Δ such that for every $c \geqslant 1$ and $\lambda \geqslant \Delta + c$, the following holds:*

- *Any intersection graph G of similarly-sized β-fat objects in $\mathbb{R}^d$ has a λ-linked partition $\mathcal{P}$ of G such that $G_{\mathcal{P}}$ has treewidth $O(n^{1-1/d})$ and maximum degree Δ.*
- *There is a $2^{O(n^{1-1/d})}$-time algorithm that, given such a graph (without a representation), returns such a partition as well as the corresponding tree decomposition.*

Moreover, the difference between λ and Δ can be made arbitrarily large.

Proof. Let $\mathcal{P}_0 = (V_1^0, \ldots, V_t^0)$ be the κ-partition obtained in Theorem 5, so that $G_{\mathcal{P}_0}$ has maximum degree Δ and treewidth $O(n^{1-1/d})$. This partition can be computed in polynomial time, and a tree decomposition of width $O(n^{1-1/d})$ in time $2^{O(n^{1-1/d})}$. We now refine each part V_i^0 of $\mathcal{P}_0$ using the following key claim. Let $g : \mathbb{N} \to \mathbb{N}$ be a function to be set later.

▷ Claim 11. Let $X \subseteq V(G)$ be a vertex set such that X admits a (unknown) clique partition of size κ. Then X can be partitioned into at most 2κ subsets, each of which either induces a clique or a $g(\kappa)$-connected subgraph, and such a partition can be found in polynomial-time.

Proof. We first construct a *separator tree* T for $G[X]$. This is a rooted tree defined as follows. If $G[X]$ has no vertex separator of size at most $g(\kappa)$, then T consists of a single leaf, which is also the root, and whose label is X. Otherwise, let $S \subseteq X$ be a vertex separator of $G[X]$ of size at most $g(\kappa)$, and let $X_1, \ldots, X_\ell$ be the vertex sets of the connected components of $G[X \setminus S]$, where $\ell \geq 2$. For each $i \in \{1, \ldots, \ell\}$, recursively construct a separator tree T_i for $G[X_i]$. Then T is obtained by creating a new root node labelled with S and making it adjacent to the roots of $T_1, \ldots, T_\ell$.

By construction, every leaf of T is labelled with a set $C \subseteq X$ such that $G[C]$ has no separator of size at most $g(\kappa)$; that is, each $G[C]$ is $g(\kappa)$-connected.

Furthermore, T has at most κ leaves. Indeed, suppose for contradiction that T had at least $\kappa + 1$ leaves $C_1, \ldots, C_{\kappa+1}$. Since the labels of distinct leaves lie in distinct connected components after removing all separators encountered on their respective root–leaf paths, they are pairwise non-adjacent. Thus picking one vertex from each C_i yields an independent set of size $\kappa + 1$, which is a contradiction with the existence of a clique partition of size κ.

Since every internal node has at least two children and the tree has at most κ leaves, it follows that T has at most $\kappa - 1$ internal nodes. Each internal node is labelled with a separator of size at most $g(\kappa)$, so the union X' of those labels satisfies $|X'| \leqslant (\kappa - 1)g(\kappa)$.

Finding a vertex separator of size at most $g(\kappa)$ can be done in time $|X|^{g(\kappa)}$. Since T contains at most $2\kappa - 1$ nodes, this operation is performed at most $2\kappa - 1$ times. Hence the total running time for constructing T is $(2\kappa - 1)\,|X|^{g(\kappa)}$. Since $X' \subseteq X$, the set X' admits a partition into at most κ cliques. As $|X'| \leqslant (\kappa - 1)g(\kappa)$, one can find such a partition by brute force in time $\kappa^{(\kappa-1)g(\kappa)}$, by enumerating all partitions of X' into κ (possibly empty) parts. Finally, note that the leaf labels of T, together with the clique partition of X', yield a partition of X into at most 2κ parts, each of them being either a clique or $g(\kappa)$-connected. ◁

Applying Claim 11 to each V_i^0 yields a refined partition of $V(G)$ into at most 2κ subsets per class, each of which induces a clique or a $g(\kappa)$-connected subgraph. Let $\mathcal{P}$ denote the resulting partition of $V(G)$, and consider the contraction $G_{\mathcal{P}}$.

Since each V_i^0 was replaced by at most 2κ subsets, $G_{\mathcal{P}}$ has maximum degree at most $\Delta' := (\Delta + 1) \cdot 2\kappa$. Moreover, its treewidth remains $O(n^{1-1/d})$, since κ is a constant.

Finally, fix $\lambda > \Delta'$ as large as desired (but constant), and fix $g(\kappa)$ large enough such that $g(\kappa) \geqslant \max(\kappa + 3, 10) \cdot 3\lambda$. Then each part of $\mathcal{P}$ is either a clique or Hamiltonian λ-linked by Lemma 8, which proves the theorem.

Note that $\mathcal{P}$ can be computed in polynomial time: the partition $\mathcal{P}_0$ is obtained in polynomial time, and by Claim 11, the refinement of each $V_i^0 \in \mathcal{P}_0$ can also be computed in polynomial time. Moreover, a tree decomposition of $G_{\mathcal{P}}$ of treewidth $O(n^{1-1/d})$ can be obtained from a tree decomposition of $G_{\mathcal{P}_0}$ by replacing each vertex V_i^0 with the vertices of its refinement in $\mathcal{P}$. Since the treewidth remains $O(n^{1-1/d})$, such a decomposition can be computed in time $2^{O(n^{1-1/d})}$. ◀

▶ Remark 12. Given a graph G, the partition $\mathcal{P} = (V_1, ..., V_t)$ of $V(G)$ obtained in Theorem 10 is a refinement of the partition $\mathcal{P}_0$ obtained from Theorem 5. By Remark 6, in any representation $\mathcal{O} = \{O_v\}_{v \in V(G)}$ of G with similarly sized β-fat objects in $\mathbb{R}^d$, two objects O_u and O_v are at distance at most 2β whenever $u, v \in V_i$ for some $1 \leqslant i \leqslant t$.

3.2 The Algorithm

We begin by adopting the technique introduced by Ito and Kadoshita, who observed that when looking for a Hamiltonian cycle in a graph G admitting a clique partition $\mathcal{P}$ such that the $\mathcal{P}$-contraction has bounded degree, it is sufficient to consider only a constant number of vertices from each clique. In particular, they proved that there always exists a Hamiltonian cycle that uses only a bounded number of edges between any two fixed cliques of the partition.

▶ **Lemma 13** ([9, Lemmas 3.2 and 3.3]). *Let G be a graph and $\mathcal{P} = (Q_1, \ldots, Q_t)$ a clique partition of G such that $G_{\mathcal{P}}$ has maximum degree Δ. There is a family $\{E_{i,j}\}_{1 \leqslant i < j \leqslant t}$ of edge sets such that the two following points hold:*

- **(i)** *for every i, j such that $1 \leqslant i < j \leqslant t$, $E_{i,j} \subseteq E(G) \cap (Q_i \times Q_j)$ and $|E_{i,j}| \leqslant 4(2\Delta - 1)^2$;*
- **(ii)** *G admits a Hamiltonian cycle if and only if it admits a Hamiltonian Cycle C that, for every i, j as above, uses at most two edges from $(Q_i \times Q_j)$, all belonging to $E_{i,j}$.*

Moreover, the sets $\{E_{i,j}\}$ can be computed in polynomial time.

In addition, we will use as a black box the FPT algorithms for Hamiltonian Path and Hamiltonian Cycle, both running on a graph G in time $2^{O(\mathrm{tw}(G))} \cdot |G|^{O(1)}$, where $\mathrm{tw}(G)$ denotes the treewidth of G.

▶ **Theorem 14** ([1, 3]). *Given a graph G together with a tree decomposition of width w, there exists an algorithm that solves Hamiltonian Cycle in time $2^{O(w)} \cdot n^{O(1)}$.*

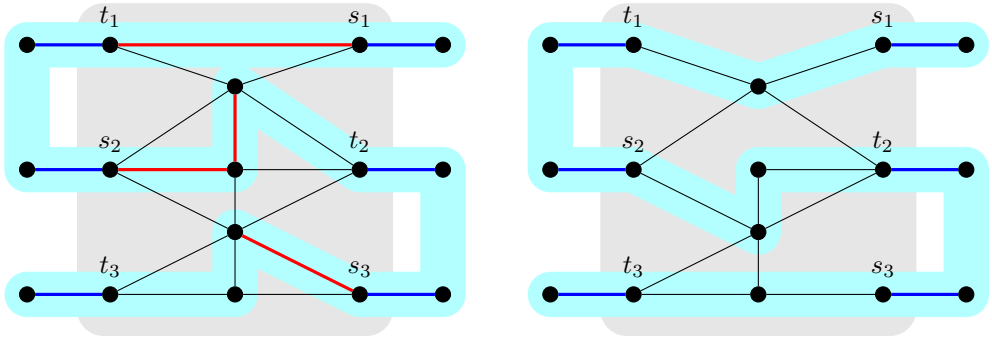


Figure 2 Illustration of the proof of Theorem 1, when turning a Hamiltonian cycle C in H (left) to a Hamiltonian cycle in G (right), focusing on a part V_i. In both figures, the grey box represents the set V_i, and the blue box represents the (partial) Hamiltonian cycle going through V_i. Left: C uses blue edges to enter and leave V_i, and uses edges from G as well as red edges inside V_i (notice that we only represented the red edges which are used by C). Right: since $G[V_i]$ is Hamiltonian-λ-linked, the (s_i, t_i)-paths which were using red edges in H can be replaced by actual paths in G.

We are now ready to prove the main result of this section.

Proof of Theorem 1. We give here an algorithm for Hamiltonian Cycle and explain at the end how to obtain that for Hamiltonian Path. According to Theorem 10, for some constants Δ and λ with $\Delta < \lambda$ we can compute a λ-linked partition $\mathcal{P} = (V_1, \ldots, V_t)$ of G and a tree decomposition of $G_{\mathcal{P}}$ in time $2^{O(n^{1-1/d})}$. In order to be able to apply Lemma 13, we define $G^\times$ as the graph obtained by turning each V_i into a clique (i.e., by adding all missing edges inside each V_i). We call the edges of $E(G^\times) \setminus E(G)$ the *red edges*.

Compute the edge sets $\{E_{i,j}\}_{1 \leqslant i < j \leqslant k}$ from Lemma 13, and call these *blue edges*. Let H be the graph obtained from $G^\times$ by removing every vertex not incident to a blue edge, except for one special vertex v_i per part V_i (if such a vertex exists). Denote by V_i^H the set of vertices of V_i that remain in H. Since $G_{\mathcal{P}}$ has maximum degree at most Δ, at most $4\Delta(2\Delta-1)^2$ vertices from each V_i are incident to blue edges, so $|V_i^H| \leqslant 4\Delta(2\Delta-1)^2 + 1$. Observe that H is a subgraph of the lexicographic product of $G_{\mathcal{P}}$ with a clique of constant size (depending on d only), hence its treewidth is $O(n^{1-1/d})$.

We now prove that G admits a Hamiltonian cycle if and only if H does.

Suppose that G has a Hamiltonian cycle. Then so does $G^\times$ as it is a supergraph of G. By Lemma 13, $G^\times$ admits a Hamiltonian cycle C that uses at most two blue edges between any V_i and V_j, that belong to $E_{i,j}$. Removing from C any vertex $v \in V_i \setminus V_i^H$ preserves Hamiltonicity in H (as V_i^H is a clique in H, and the neighbors of v in C must belong to V_i, otherwise v would be adjacent to a blue edge), hence H has a Hamiltonian cycle.

Conversely, suppose that H has a Hamiltonian cycle C. We reconstruct a Hamiltonian cycle of G as follows. For each $i \in [t]$, if $G[V_i]$ is a clique, we can insert the missing vertices of $V_i \setminus V_i^H$ along C. Indeed, if V_i^H contains a special vertex v_i, then C must contain an edge uv_i with $u \in V_i$, and thus all vertices of $V_i \setminus V_i^H$ can be inserted between u and v_i in C. If V_i^H do not contain a special vertex, then $V_i^H = V_i$ and C already contain all vertices of V_i.

If instead $G[V_i]$ is Hamiltonian-λ-linked, fix an orientation of C and let $s_1, \ldots, s_m$ be the vertices of V_i where C enters V_i, and $t_1, \ldots, t_m$ where it exits. Observe that $2m \leqslant 2\Delta$ as one can map each s_j (resp. t_j) to the unique blue edge that C uses to enter (resp. exit) V_i^H, and C uses at most 2Δ blue edges adjacent to V_i^H (at most two blue edges per adjacent part of V_i in $G_{\mathcal{P}}$). Since $G[V_i]$ is Hamiltonian-λ-linked with $\lambda > \Delta$, there exist m vertex-disjoint paths in $G[V_i]$ covering V_i and connecting each s_j to t_j. Replacing the subpaths of C within V_i by these paths yields a Hamiltonian cycle of G. See Figure 2 for an illustration.

To sum up, we reduced the problem to that of deciding whether H has a Hamiltonian cycle. Recall that H has treewidth $O(n^{1-1/d})$. This can be done in time $2^{O(n^{1-1/d})}$ using Theorem 14 on the tree decomposition of H computed above. Each local reconstruction inside a part V_i can be carried out using the algorithm of Lemma 8 in polynomial time.

With minor adjustments, Lemma 13 also applies to the Hamiltonian Path problem. More precisely, if G admits a Hamiltonian path, one can construct a graph G' by adding a special edge between the two endpoints of that path. The resulting graph G' admits a clique partition of bounded degree, and thus Lemma 13 applies to it. The Hamiltonian cycle obtained in G' may include the special edge; removing this edge then yields a Hamiltonian path in the original graph G. Then, the very same proof carries over to Hamiltonian Path, yielding an algorithm with identical running time. ◀

4 A Subexponential FPT algorithm for Long Path

This section is dedicated to the proof of Theorem 2. We first present a low-treewidth pattern covering technique adapted to geometric graphs, and then solve the Long Path problem using both this technique with λ-linked partitions.

4.1 Low-Treewidth Pattern Covering

We say that a graph class $\mathcal{G}$ has *growth* at most f if for every $G \in \mathcal{G}$ and $v \in V(G)$, the ball $B_G(v, r)$ contains at most $f(r)$ vertices. The next lemma shows that intersection graphs of similarly sized fat objects of bounded maximum degree have growth at most a polynomial function.

▶ **Lemma 15.** *For every constants $d \geqslant 2$, $\beta \geqslant 1$ and $\Delta > 0$, any intersection graph of similarly sized β-fat objects in $\mathbb{R}^d$ has growth $O(r^d)$.*

Proof. Let $\{O_v\}_{v \in V(G)}$ be a representation of G with similarly sized β-fat objects in $\mathbb{R}^d$. Let $v \in V(G)$. Observe than since the object associated to every neighbor of v intersects O_v, and these objects are β-fat, there is a d-dimensional hypercube of side length 6β that delimits a region containing the objects of v and all its neighbors. More generally for any integer $r \geqslant 1$, all the vertices at distance at most r from v in G have their objects in the region of $\mathbb{R}^d$ enclosed by a d-dimensional hypercube of side length $c_\beta r$, for some constant c_β depending only on β.

This hypercube can be partitioned into at most $c_\beta^d r^d$ smaller hypercubes of unit side length. Since the objects are β-fat, the object O_u associated to each $u \in V(G)$ contains a ball of radius 1. Let us denote by z_u the center of such a ball. Observe that if a smaller hypercube contains z_u and $z_{u'}$ for some $u, u' \in V(G)$, then O_u and $O_{u'}$ intersect and so u and u' are adjacent. Hence each such smaller hypercube contains at most $\Delta + 1$ points of the form z_u for $u \in V(G)$ (whose associated vertices form a clique of order at most $\Delta + 1$). Therefore, the ball $B_G(v, r)$ contains at most $c_\beta^d(\Delta + 1) \cdot r^d$ vertices, as claimed. ◀

▶ Remark 16. As a consequence of the previous lemma and the algorithm of [12], there exists a randomized algorithm that solves Long Path in time $2^{O(n^{1-1/(d+1)})}$ in intersection graphs of similarly sized fat objects in $\mathbb{R}^d$, provided that the maximum degree is constant.

We will use the following lemma showing that in graphs of polynomial growth, one can select a subset of vertices inducing connected components of radius $O(k \log k)$ and that has good probability of containing an unknown subset $X \subseteq V(G)$ of size at most k.

▶ **Lemma 17** ([12, Lemma 2.1]). *Let $\mathcal{G}$ be a graph class of growth $O(r^\delta)$. There exists a constant $c > 0$ and a polynomial-time randomized algorithm that, given $G \in \mathcal{G}$ and an integer $k \geqslant 4$, outputs a subset $A \subseteq V(G)$ satisfying:*

1. *every connected component of $G[A]$ has radius at most $ck \log k$;*
2. *for every set $X \subseteq V(G)$ of size at most k, the probability that $X \subseteq A$ is at least $\frac{17}{256}$.*

We then give a low-treewidth pattern covering lemma for intersection graphs of similarly sized fat objects and bounded maximum degree, with diameter polynomial in k. The proof, available in the long version, follows essentially the same clustering procedure as in [12]. Roughly speaking, the idea is to construct a collection of disjoint *clusters* in the graph, where each cluster induces a subgraph of small treewidth, and then add a small number of vertices that are adjacent to these clusters in order to ensure connectivity between them. The main difference with [12] lies in the analysis of the treewidth of the clusters, which can be improved in the case of geometric graphs.

▶ **Lemma 18** (✂). *For every constants $d \geqslant 2$, $\beta \geqslant 1$, $\Delta \geqslant 1$ and $c > 0$, there exists a constant $c' > 0$ and a polynomial-time randomized algorithm that, given $k \geqslant 4$ and a connected intersection graph G of similarly-sized β-fat objects in $\mathbb{R}^d$ of maximum degree Δ and radius at most $ck \log k$, outputs a subset $A \subseteq V(G)$ such that:*

1. *the treewidth of $G[A]$ is $O(k^{1-1/d} \log k)$;*
2. *for every set $X \subseteq V(G)$ of size at most k, the probability that $X \subseteq A$ is at least $2^{-c'(|X|+k)k^{-1/d}\log^2 k}$.*

4.2 Robust Subexponential FPT algorithm for Long Path

Recall that there exists an algorithm solving Long Path and Long Cycle in subexponential FPT time $2^{O(\sqrt{k})} \cdot n^{O(1)}$ in unit-disk graphs [16], and that this running time is tight under the Exponential Time Hypothesis (ETH). The approach to obtain this algorithm is very similar that used to obtain a subexponential algorithm running in time $2^{O(\sqrt{n})}$.

The key idea is to view G as a *clique-grid* graph. Informally, a clique-grid graph is a graph equipped with a mapping $f : V(G) \to [t]^2$ such that, for all $(i, j) \in [t]^2$, the preimage $f^{-1}(i, j)$ induces a clique, and edges between vertices in $f^{-1}(i, j)$ and $f^{-1}(i', j')$ exist only if the distance between the cells (i, j) and (i', j') is less than a fixed constant. The algorithm can then be seen as consisting of two main steps:

- **(i)** marking a constant number of vertices in each clique such that the path only traverses the cliques via these marked vertices, and
- **(ii)** applying bidimensionality techniques to solve a weighted version of Long Path on the subgraph induced by the marked vertices in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$.

The mapping f is straightforward to obtain when a geometric representation of the underlying objects is available. However, at the time of writing, it is not know how to do so without the representation. Importantly, the marking scheme does not fundamentally rely on G being a clique-grid graph, but only on the existence of a clique partition $\mathcal{P}$ such that $G_{\mathcal{P}}$ has bounded degree. The following lemma is a straightforward rewriting of the marking scheme of [16]. It can be viewed as a generalization of Lemma 13, which additionally handles the case where the path is not necessarily Hamiltonian.

▶ **Lemma 19** (Reformulation of [16, Lemma 14]). *Let G be a graph and $\mathcal{P} = (Q_1, \ldots, Q_t)$ a clique partition of G such that the $\mathcal{P}$-contraction $G_{\mathcal{P}}$ has degree at most Δ. There exists a family of vertex subsets $\{\mathsf{M}(i)\}_{1 \leqslant i \leqslant t}$ with $\mathsf{M}(i) \subseteq Q_i$ for every $i \in \{1, \ldots, t\}$ and satisfying*

the following property. If G contains a path with k vertices, then it also contains a path P with endpoints x, y such that, for every $1 \leqslant i \leqslant t$,

1. *either $V(P) \subseteq Q_i$, or*
2. *$V(P) \cap (Q_i \setminus \mathsf{M}(i)) = \emptyset$, or*
3. *there exist distinct vertices $u, v \in (V(P) \cap \mathsf{M}(i)) \cup (\{x, y\} \cap Q_i)$ such that the set of internal vertices of the subpath of P between u and v is precisely*

$$(V(P) \cap Q_i) \setminus (\mathsf{M}(i) \cup \{u, v\}).$$

In addition, M can be computed in polynomial time, and $|\mathsf{M}(i)| = \Delta^{O(1)}$ for any $1 \leqslant i \leqslant t$.

In the Weighted Long Path problem, the input is a graph G, a weight function $w : V(G) \to \mathbb{N}$ and an integer W, and the goal is to find a path $(u_1, \ldots, u_q)$ in G ($q > 0$), such that $\sum_{i \in [q]} w(u_i) \geqslant W$. Adapting the classical dynamic programming algorithm for Long Path on graphs with bounded treewidth [2], we can obtain the following result.

▶ **Proposition 20** (see [2]). *Weighted Long Path is solvable in time $2^{O(w)} \cdot n^{O(1)}$ on n-vertex graphs of treewidth at most w.*

We are now ready to prove the main result of this section.

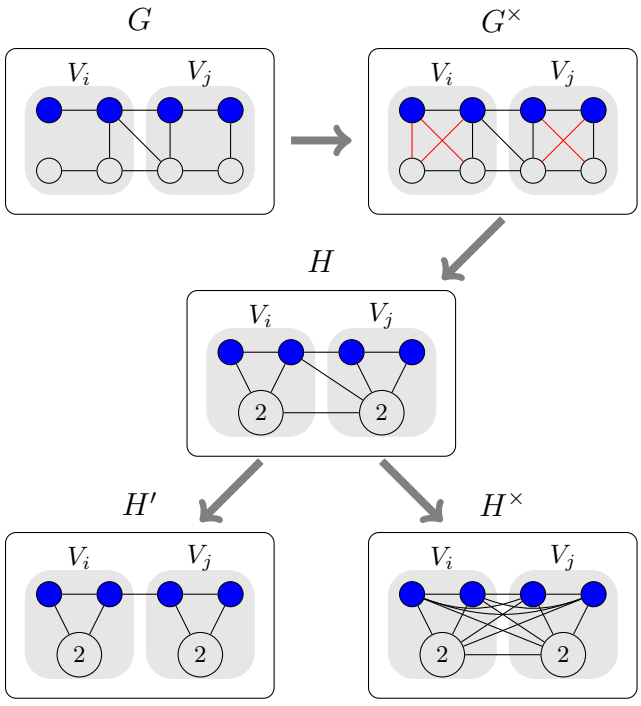


Figure 3 Illustration of the five graphs of the proof of Theorem 2.

Proof of Theorem 2. We begin by computing, in polynomial time, the λ-linked partition $\mathcal{P} = (V_1, \ldots, V_t)$ guaranteed by Theorem 10, such that $G_{\mathcal{P}}$ has maximum degree Δ, for some constants λ and Δ such that $\Delta < \lambda$. Let $G^\times$ denote the graph obtained from G by turning each V_i into a clique, that is, by adding all missing edges inside each V_i. Again, the edges in $E(G^\times) \setminus E(G)$ are called *red edges*.

We apply Lemma 19 to the pair $(G^\times, \mathcal{P})$, yielding the marking function M defined on $V(G^\times)$. From this, we build the weighted graph H as follows: for each $1 \leqslant i \leqslant t$, let $U_i := V_i \setminus \mathsf{M}(i)$, and contract all vertices in U_i into a single vertex v_i with weight $w_i := |U_i|$. Whenever $U_i \neq \emptyset$, define $Q_i := \mathsf{M}(i) \cup \{v_i\}$. If $U_i = \emptyset$, do not create vertex v_i and define $Q_i := \mathsf{M}(i)$. All remaining vertices (the marked ones) receive weight 1. Note that Q_i is a clique in H for all $1 \leqslant i \leqslant t$.

We now define two auxiliary graphs derived from H:

1. H' is obtained from H by deleting every edge $v_i u$ with $u \in V_j$ and $j \neq i$.
2. $H^\times$ is obtained from H by adding, for every $1 \leqslant i < j \leqslant t$, all edges between Q_i and Q_j whenever there exists an edge $uv \in E(H')$ with $u \in Q_i$ and $v \in Q_j$. Note that, for any $u, v \in Q_i$, u and v are *true twins* in $H^\times$, meaning that $N_{H^\times}[u] = N_{H^\times}[v]$. Moreover, H' is a subgraph of $H^\times$.

See Figure 3 for an illustration of the graphs G, $G^\times$, H, H' and $H^\times$. We first prove that $H^\times$ is the intersection graph of similarly sized fat objects in $\mathbb{R}^d$.

▷ Claim 21. There exist constants β' and Δ' such that $H^\times$ is the an intersection graph of similarly-sized β'-fat objects in $\mathbb{R}^d$, and has maximum degree at most Δ'.

Proof. Let $\mathcal{O} = \{O_v\}_{v \in V(G)}$ be a geometric representation of G with d-dimensional similarly sized β-fat objects. For each i, let $O_i = \bigcup_{v \in V_i} O_v$. Since V_i is λ-linked, the diameter of O_i is bounded by $O(\beta)$ by Remark 12, and O_i contains a ball of radius 1 (as each O_v does). We represent every vertex $u \in Q_i$ by the same object O_i.

Note that for each $1 \leqslant i \leqslant t$, the objects representing Q_i pairwise intersect, since they are all represented by the same geometric object. Moreover, for any $1 \leqslant i < j \leqslant t$, the objects O_i and O_j intersect whenever there exist vertices $u \in V_i$ and $v \in V_j$ such that O_u and O_v intersect. It follows that $H^\times$ is indeed the intersection graph of $\{O_u : u \in V(H^\times)\}$. In addition, its degree is bounded by $\Delta' = \Delta^{O(1)}$, as each O_i intersects only a bounded number of other regions corresponding to adjacent parts in $G_{\mathcal{P}}$. ◁

We now show that G has a long path if and only if H' does.

▷ Claim 22. G has a path on at least k vertices if and only if H' has a weighted path of total weight at least k. Moreover, given such a path in H', one can construct a path in G with at least the same number of vertices in polynomial time.

Proof. Suppose that G contains a path P on k vertices with endpoints x and y. Since G is a subgraph of $G^\times$, P is also a path in $G^\times$. By Lemma 19, P can be modified into a path P' such that for each i, P' contains all vertices in $(V(P) \cap V_i) \setminus \mathsf{M}(i)$ consecutively along P', and we call P_i such a subpath. By contracting each such P_i into a single vertex v_i, we obtain a path P_H in H' whose total weight equals $|V(P)| \geqslant k$.

Conversely, let P_H be a weighted path in H' of total weight at least k. For each contracted vertex v_i of weight w_i, we expand it into an arbitrary path of w_i distinct vertices from U_i. The two vertices (if they exist) adjacent to v_i in P_H must belong to $\mathsf{M}(i)$ by definition of H', so the resulting expanded path P' exists in $G^\times$ and has at least k vertices.

It remains to transform the path P' in $G^\times$ into a path in G. Currently, P' may use red edges inside some V_i that is not a clique in G, but each $G[V_i]$ is Hamiltonian λ-linked. Fix i such that $G[V_i]$ is Hamiltonian λ-linked, and let $s_1, \ldots, s_m$ be the vertices of V_i where P' enters V_i, and $t_1, \ldots, t_m$ the vertices where it exits. Since P' must enter and exit through marked vertices, we have $m = \Delta^{O(1)}$. If λ is chosen so that $\lambda > m$, there exist m vertex-disjoint paths $\{P_{i,j}\}_{1 \leqslant j \leqslant m}$ covering all vertices of V_i, each connecting s_j to t_j. Replacing each subpath between s_j and t_j in P' by $P_{i,j}$ yields a new path with at least the same

number of vertices, and without using red edges. Performing this replacement for every part V_i such that $G[V_i]$ is Hamiltonian λ-linked yields a path on at least k vertices in G. This construction can be carried out in polynomial time by Lemma 8. ◁

We now describe the algorithm. Suppose that G has a path P with k vertices. We apply successively Lemma 17 and Lemma 18 to $H^\times$ (which, by the previous claim, is the intersection of similarly-sized fat objects in $\mathbb{R}^d$, and has bounded maximum degree) to obtain a set $A \subseteq V(H^\times)$ such that:

1. the treewidth of $H^\times[A]$ is $O(k^{1-1/d} \log k)$;
2. with probability at least $2^{-O(k^{1-1/d} \log^2 k)}$, we have $V(P) \subseteq A$.

Since H' is a subgraph of $H^\times$, $H'[A]$ also has treewidth $O(k^{1-1/d} \log k)$. Then, one can find a path on k vertices in $H'[A]$ using Proposition 20 in time $2^{O(k^{1-1/d} \log k)} \cdot n^{O(1)}$. Repeating this process $2^{O(k^{1-1/d} \log^2 k)}$ times ensures constant success probability. If a path on k vertices is found in H', we construct in polynomial time a path on k vertices of G using Claim 22. If no such path is found after all repetitions, the algorithm reports that G has no path on k vertices. ◀

5 Conclusion

In this paper we have presented a robust algorithm for solving Hamiltonian Path and Hamiltonian Cycle in intersection graphs of similarly sized fat objects in $\mathbb{R}^d$ in time $2^{O(n^{1-1/d})}$, which is tight under ETH. In addition, we extended our method to obtain a robust randomized FPT algorithm for Long Path, running in time $2^{O(k^{1-1/d} \log^2 k)}$. We conclude with two open questions:

- Is it possible to remove the $\log^2 k$ factor in the exponent of our algorithm for Long Path, and to make the algorithm deterministic?
- Our method does not directly yield a robust subexponential FPT algorithm for Long Cycle. Is it possible to obtain one using a different approach?

References

1 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015. doi:10.1016/J.IC.2014.12.008.

2 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 1 edition, 2015. doi:10.1007/978-3-319-21275-3.

3 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Joham MM van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 150–159. IEEE, 2011.

4 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for ETH-tight algorithms and lower bounds in geometric intersection graphs. *SIAM Journal on Computing*, 49(6):1291–1331, 2020.

5 Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing*, 34(6):1302–1323, 2005. doi:10.1137/S0097539702402676.

6 Fedor V. Fomin, Petr A Golovach, Nikola Jedličková, Jan Kratochvíl, Danil Sagunov, and Kirill Simonov. Path cover, hamiltonicity, and independence number: An FPT perspective. *arXiv preprint*, 2024. arXiv:2403.05943.

7 Fedor V. Fomin, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Subexponential parameterized algorithms for planar and apex-minor-free graphs via low treewidth pattern covering. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 515–524. IEEE, 2016. doi:10.1109/FOCS.2016.62.

8 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, hitting and packing cycles in subexponential time on unit disk graphs. *Discrete & Computational Geometry*, 62(4):879–911, 2019. doi:10.1007/S00454-018-00054-X.

9 Hiro Ito and Masakazu Kadoshita. Tractability and intractability of problems on unit disk graphs parameterized by domain area. In *Proceedings of the 9th International Symposium on Operations Research and Its Applications (ISORA10)*, pages 120–127, 2010.

10 Ross J Kang and Tobias Müller. Sphere and dot product representations of graphs. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 308–314, 2011. doi:10.1145/1998196.1998249.

11 Tomohiro Koana, Nidhi Purohit, and Kirill Simonov. Subexponential Algorithms for Clique Cover on Unit Disk and Unit Ball Graphs. In Édouard Bonnet and Paweł Rzążewski, editors, *19th International Symposium on Parameterized and Exact Computation (IPEC 2024)*, volume 321 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.IPEC.2024.10.

12 Dániel Marx and Marcin Pilipczuk. Subexponential Parameterized Algorithms for Graphs of Polynomial Growth. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 59:1–59:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2017.59.

13 Imran A Pirwani and Mohammad R. Salavatipour. A weakly robust PTAS for minimum clique partition in unit disk graphs. *Algorithmica*, 62(3):1050–1072, 2012. doi:10.1007/S00453-011-9503-8.

14 Frank van den Eijkhof, Hans L Bodlaender, and MC Arie Koster. Safe reduction rules for weighted treewidth. *Algorithmica*, 47(2):139–158, 2007. doi:10.1007/S00453-006-1226-X.

15 Jie Xue and Meirav Zehavi. Parameterized algorithms on geometric intersection graphs. *Computer Science Review*, 58:100796, 2025. doi:10.1016/J.COSREV.2025.100796.

16 Meirav Zehavi, Fedor Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. ETH-tight algorithms for long path and cycle on unit disk graphs. *Journal of Computational Geometry*, 12(2):126–148, April 2022. doi:10.20382/JOCG.V12I2A6.

Mapping Chemical Space: Topological Data Analysis of Chemical Latent Space with Mapper

Dhruv Meduri
University of Utah, Salt Lake City, UT, USA

Chuan-Shen Hu
Nanyang Technological University, Singapore

Cong Shen
National University of Singapore, Singapore

Kelin Xia
Nanyang Technological University, Singapore

Bei Wang[1]
University of Utah, Salt Lake City, UT, USA

Abstract

The vast chemical space, encompassing virtually innumerable molecules and materials, presents both immense opportunities and significant challenges. The design and discovery of novel drugs and functional materials may be viewed as a search within this space; however, the sheer scale of potential candidates renders exhaustive exploration infeasible. To address this, we introduce *Chemical Mapper*, a framework that integrates topological data analysis with deep learning to enable the visual exploration and analysis of chemical latent spaces. At its core, *Chemical Mapper* employs mapper, a widely used tool in topological data analysis, to investigate the organizational principles of chemical latent spaces defined by molecular representations learned by geometric deep learning models. In doing so, *Chemical Mapper* not only highlights groups of molecular representations but also uncovers the relationships among them through linkages and branching structures. Our results show that *Chemical Mapper* reveals intrinsic patterns associated with molecular scaffolds, functional groups, and chemical properties, as well as the structural and functional evolutions of the molecules.

2012 ACM Subject Classification Applied computing → Chemistry; Mathematics of computing → Algebraic topology

Keywords and phrases Practice of computational topology, topological data analysis, applications in chemistry, mapper algorithm, high-dimensional data analysis, chemical spaces, geometric deep learning, latent space geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.78

Supplementary Material *Dataset (Source code is also added)*: `https://doi.org/10.5281/zenodo.19241865`

Funding *Dhruv Meduri*: NSF IIS-2205418 and DMS-2134223.
Kelin Xia: Tier 1 grant RG16/23 and Tier 2 grant MOE-T2EP20125-0004.
Bei Wang: NSF IIS-2205418 and DMS-2134223.

1 Introduction

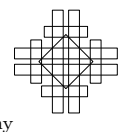
Navigating chemical space, which encompasses the immense number of compounds formed by all possible atomic combinations, is both challenging and full of promise [10, 24, 42, 41]. For example, the GDB-17 dataset, which includes molecules composed of up to 17 atoms

[1] Corresponding author



© Dhruv Meduri, Chuan-Shen Hu, Cong Shen, Kelin Xia, and Bei Wang;
licensed under Creative Commons License CC-BY 4.0





of C, N, O, S, and halogens, already enumerates 166.4 billion structures [41]. Estimates suggest that more than 10^{60} molecules could satisfy Lipinski's rule-of-five and thus represent potential drug candidates. Even restricting attention to drug candidates of up to 30 atoms yields a chemical space of about 10^{20}–10^{24} compounds. This staggering diversity provides an effectively unlimited reservoir of opportunities for novel drugs, functional materials, and green technologies [24, 42]. Indeed, chemical space is widely regarded as the ultimate frontier for drug discovery, as all approved drugs originate from exhaustive screening and refinement of candidates within this vast molecular universe [42].

Understanding and navigating the vast landscape of chemical space is essential for advancing drug discovery, materials science, and chemical engineering [3, 13, 24, 29, 44, 50, 52]. Broadly, there are two principal approaches for the visualization and analysis of chemical space: *global clustering* and *local topology*.

Global clustering models are frequently built on dimensionality reduction techniques [14, 23]. For instance, the *Chemical Space Project* [41] employs PCA [34] to illustrate chemical diversity, whereas *ChemPlot* [46], a Python library for chemical space visualization, supports PCA, t-SNE [47], as well as UMAP [28]. *Drug Discovery Maps* (DDM) [20] applies t-SNE to visualize the molecular similarity of experimental drugs in chemical space. More recently, AI-driven approaches have been adopted to study the chemical latent space, which embeds molecule structures into a mathematical space defined by molecular features. This chemical latent space captures structural diversity within chemical libraries and supports the exploration of broader chemical space for designing novel drug candidates [32]. Variational autoencoders (VAEs) have been widely used for both learning [15] and constructing chemical latent spaces [32]. The overarching idea is to transform chemical structures into low-dimensional or latent representations that preserve key structural and functional properties, thereby enabling visualization, clustering, and analysis. A variety of clustering methods [18, 19, 33] have been applied in this context. For example, Cheng et al. [9] used unsupervised clustering with Gaussian mixture models (GMMs) to partition chemical space, while Hadipour et al. [16] applied K-means [26] and BIRCH [51] – a scalable hierarchical clustering algorithm – to group VAE-based molecular representations. A recent extension, BitBIRCH [35], was shown to cluster 450,000 molecules in around 2.2 minutes.

In contrast, local topology models highlight connections at the level of individual compounds [25, 27, 36, 43]. For example, interaction networks among proteins, ligands, RNAs, DNAs, and other molecules are widely used to model chemical space. Beyond direct interactions, molecular graphs can also be constructed based on structural similarity. While these graph models can also support visualization and clustering, their key strength lies in preserving local molecular connectivity, which is particularly valuable for molecular search, optimization, and modification under structural or functional constraints.

The mapper construction [45] is a central tool in topological data analysis, offering a topological summary that balances global clustering with local structure. As a discrete analogue of the Reeb graph [2, 40], mapper provides a versatile framework for exploring and analyzing high-dimensional data [4, 7, 30]. Unlike conventional clustering methods, it not only identifies clusters but also encodes their interrelationships. Crucially, mapper preserves the intrinsic topology of the data space, enabling the detection of subtle structural features such as branches, chains, and loops that are often overlooked by dimensionality reduction techniques. In its simplest form, the *mapper graph* represents clusters as nodes and their overlaps as edges. Recently, mapper graphs have emerged as powerful tools for interpreting the structure of deep learning activation spaces [5, 37, 38, 54, 55] and exploring the topology of word embeddings [39].

We present *Chemical Mapper*, a mapper framework for the visual exploration and analysis of chemical latent space. The central idea is to employ mapper graphs to investigate the organizational principles of chemical latent spaces learned by deep learning models. Specifically, we focus on chemical latent space defined by molecular representations learned from geometric deep learning models such as GEM (Geometry-Enhanced Molecular representation learning, also referred to as the Geometry Embedding Model) [13], GraphMAE [17], and Mole-BERT [48]. From these latent representations, we construct a mapper graph in which each node denotes a cluster of molecule representations and each edge encodes the overlap between clusters. Our analysis shows that edges in the mapper graph capture shared motifs, substructures, or scaffold types, while topological patterns such as branches and chains reveal systematic variations in functional groups and chemical properties.

In addition, *Chemical Mapper* enables the characterization of how molecular representations evolve during the pre-training and fine-tuning phases of deep learning models. By comparing mapper graphs across training stages, we trace the progression of molecular representations and assess their ability to generalize chemical properties. This provides new insight into the latent chemical structures learned by deep models.

Overall, this work advances topology-driven molecular analysis, highlighting the value of mapper graphs for both chemical space exploration and the interpretation of deep learning models. Our findings emphasize the importance of integrating topological data analysis with machine learning to achieve structured, interpretable molecular representations, thereby paving the way for future developments in computational chemistry, materials discovery, and AI-driven chemical modeling.

The source code and datasets associated with *Chemical Mapper* are publicly available at `https://doi.org/10.5281/zenodo.19241865`, along with supplementary videos that demonstrate its interactive exploration capabilities.

2 Materials and Methods

2.1 Datasets

To demonstrate the adaptability of our *Chemical Mapper* framework in exploring diverse chemical latent spaces, we employ three chemical datasets in this study: (1) PubChem, a large-scale molecular repository; (2) a toxicity dataset containing rat oral acute toxicity data; and (3) a water solubility dataset with $logS$ values that quantify how readily compounds dissolve in water.

First, to illustrate the representational capability of mapper in capturing molecular distributions, we focus on a randomly selected subset of two million chemical structures from the PubChem database [21], a comprehensive repository of small molecules. Second, to highlight the ability of *Chemical Mapper* to capture both molecular structures and associated physicochemical properties, we incorporate two additional datasets from Deep-PK [31], which focus on toxicity and water solubility. The toxicity dataset contains 10,130 molecules and measures acute oral toxicity in rats, typically expressed as the lethal dose for 50% of test animals (LD_{50}). The water solubility dataset comprises 9,964 molecules, each labeled with the logarithm of water solubility (logS) at 20-25°C, expressed in log mol/L.

2.2 AI-Enhanced Chemical Structural Representations

For structural analyses using *Chemical Mapper* (Section 3.2), we employ GEM [13] as the primary model for generating molecular structural representations. GEM utilizes a geometry-based graph neural network (GNN) architecture enhanced with self-supervised learning

strategies designed to capture fine-grained molecular geometry. Molecular representations are produced using a pre-trained GEM model for two million molecules, each represented as a 32-dimensional real-valued vector.

To further examine the training behavior of other geometric deep learning models, we also evaluate GraphMAE [17] and Mole-BERT [48] under pre-trained and fine-tuned conditions (Section 3.3). GraphMAE, inspired by masked autoencoding techniques in natural language processing, learns effective graph representations through self-supervised learning by masking portions of node features and training the network to reconstruct them. In contrast, Mole-BERT adopts a transformer-based architecture to capture rich structural and chemical information. Drawing inspiration from BERT, it employs masking strategies to model contextual relationships within molecular graphs, enabling more flexible and comprehensive molecular feature learning.

2.3 Mapper Algorithm

The mapper construction is a powerful topological data analysis technique that captures the intrinsic shape of high-dimensional datasets [45]. Its 1D form, the mapper graph, provides a simplified graphical summary of complex point clouds, where nodes represent clusters and edges capture overlapping relationships among them. When applied to chemical space, the mapper graph enables hierarchical exploration of molecular scaffolds, functional groups, and properties, revealing meaningful topological patterns and chemical insights.

The mapper algorithm takes as input a high-dimensional point cloud $\mathbb{X}$ equipped with a real-valued function $f : \mathbb{X} \to \mathbb{R}$, referred to as a *filter function*. It produces a graphical summary of $\mathbb{X}$ in the form of a *mapper graph*, where each node represents a cluster interpreted as a topological neighborhood, and an edge connects two nodes if their corresponding neighborhoods overlap.

We illustrate the construction of a mapper graph using a 1D point cloud $\mathbb{X}$ sampled from the letter $\mathcal{Y}$, rendered in a mathematical script font. As shown in Figure 1 (left), $\mathbb{X}$ is equipped with a height function f, which serves as the filter function. To obtain a topological summary of $\mathbb{X}$, we begin with a finite cover $\mathcal{V} = \{V_1, V_2, \dots\}$ of $f(\mathbb{X})$, formed by a set of partially overlapping intervals such that $f(\mathbb{X}) \subseteq \bigcup_j V_j$ (see Figure 1, middle). For each interval V_j, we consider the clusters induced by points in $f^{-1}(V_j)$; these clusters form a finite cover $\mathcal{U} = \{U_1, U_2, \dots\}$ of $\mathbb{X}$ such that $\mathbb{X} \subseteq \bigcup_j U_j$, highlighted by rectangles. For example, points in $f^{-1}(V_1)$ form two clusters, U_1 and U_2, while those in $f^{-1}(V_2)$ yield four. The 1D *nerve* of $\mathcal{U}$ defines the *mapper graph* in Figure 1 (right), where each cluster U_i corresponds to a node i, and an edge connects nodes i and j whenever clusters U_i and U_j have a nonempty intersection. For example, an edge connects the nodes representing clusters U_2 and U_3. The resulting mapper graph captures key topological features of the letter $\mathcal{Y}$, including its branching and looping structures.

In our study, we use the l^2-norm as the filter function, which quantifies how strongly a model is activated by an input molecule and has been shown to yield meaningful insights when analyzing latent representations in deep learning [38, 39]. Given a fixed filter function, the mapper graph construction is determined by: (1) the number of intervals and their overlap, which control the resolution of the cover; and (2) the parameters of the clustering algorithm. The overlap among cover elements is crucial for maintaining connectivity among topological neighborhoods at a given resolution, thereby revealing both local and global topological structure. In practice, we employ DBSCAN [12] for clustering, with parameters ϵ and `minPts`, where ϵ defines the neighborhood radius determining how close points must be to belong to the same cluster, and `minPts` specifies the minimum number of points required to form a dense cluster. See Section 4.1 for a discussion on parameter turning.

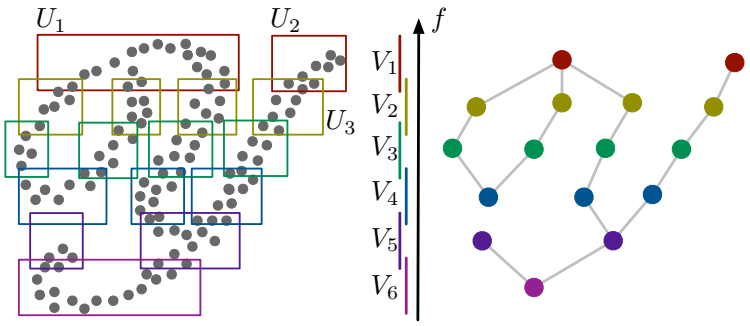


Figure 1 A simple illustration of mapper graph construction. Left: a point cloud $\mathbb{X}$ is equipped with a height function f. Middle: a cover $\mathcal{U} = \{U_1, U_2 \dots\}$ of $\mathbb{X}$ is induced by a cover $\mathcal{V} = \{V_1, V_2, \dots\}$ of $f(\mathbb{X})$ consisting of 6 intervals with 25% overlap. Right: the 1D nerve of $\mathcal{U}$ yields the mapper graph.

3 Results

3.1 Topological Data Analysis with Mapper

The mapper construction offers a key advantage in its ability to integrate topological structure into clustering analysis. As an initial study, we compare t-SNE [47] and mapper [45] for visualizing the chemical space of molecular representations.

Our study uses two million chemical structures from the PubChem database [22], comprising ten distinct scaffolds (i.e., core molecular frameworks), each represented by 200,000 molecules. Each chemical graph is converted into a 32-dimensional feature vector using the pre-trained GEM model. Detailed experimental settings are provided in Section 2.

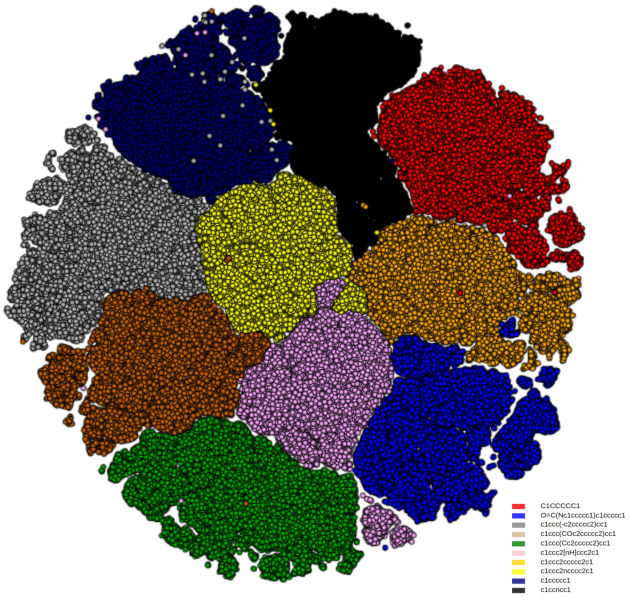


Figure 2 Application of t-SNE to two million chemical structures from the PubChem database. The embedding reveals scaffold-based clusters but offers limited insight into both intra- and inter-cluster relationships.

In the t-SNE visualization shown in Figure 2, each data point is colored by its chemical scaffold. In contrast, the mapper graphs used throughout this work represent each node as a group of molecular representations, visualized as a pie chart whose color proportions reflect the distribution of scaffolds within the node (see Figures 10 and 11), except in Section 3.2.3. Nodes rendered in a single color indicate high purity, meaning all molecules share the same scaffold (e.g., the mapper graph in Figure 5 contains only one impure node out of 5,104 total nodes). Although nodes are displayed with a uniform size by default, they may contain varying numbers of molecules, and their sizes can be adjusted if needed. The corresponding scaffold colormap, along with representative molecular structures, is shown in Figure 3.

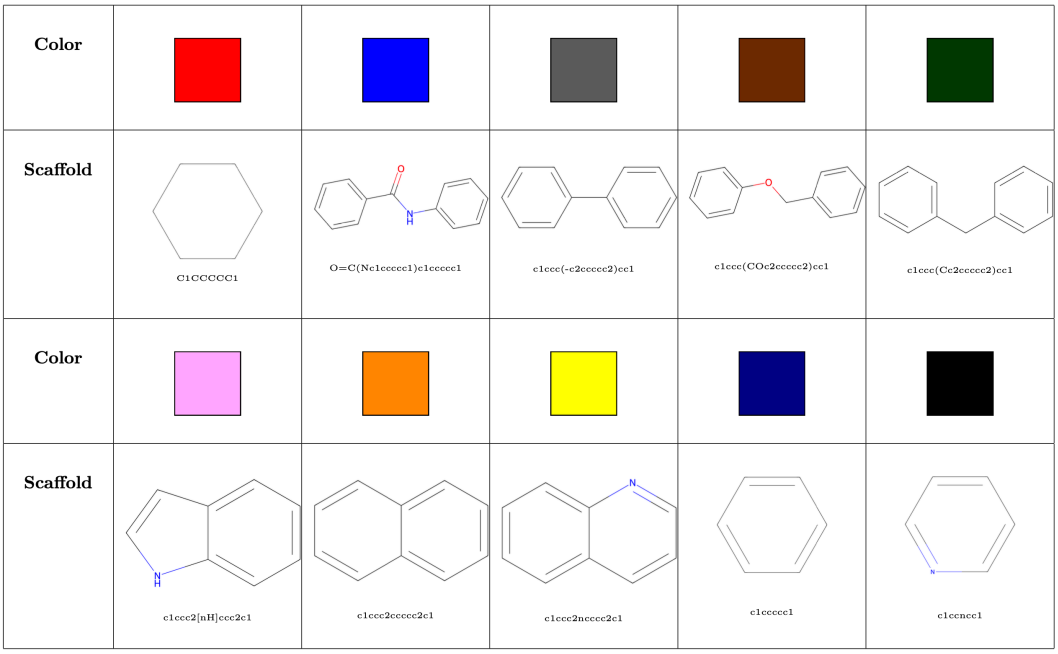


Figure 3 Scaffold colormap with representative molecular structures.

While t-SNE organizes molecular representations into ten well-separated clusters, it does not capture relationships among them. In contrast, the mapper graph in Figure 5 reveals both inter- and intra-cluster connectivity and exposes finer-grained topological features – such as branches, chains, and loops – that capture meaningful structural distinctions within the chemical space. By capturing both local and global organization, the mapper graph offers a more comprehensive view of molecular structure, surpassing the representational capacity of conventional clustering methods. In subsequent sections, our results demonstrate that the mapper graph effectively uncovers topological organization within the chemical space, revealing clusters and their relationships and providing deeper insights into molecular structure and function (see Sections 3.2 and 3.3 for details).

3.2 Mapping GEM Chemical Latent Space with Mapper

We integrate molecular representations learned from geometric deep learning models into the mapper algorithm to create an exploratory framework, *Chemical Mapper*. As illustrated in Figure 4, the workflow begins with molecular datasets from which latent representations are extracted using deep learning models. These representations form a high-dimensional point cloud that is analyzed using the mapper algorithm. The resulting mapper graph

captures relationships among scaffolds, functional groups, and molecular properties. Downstream analyses leverage this graph to reveal structural patterns and property distributions, highlighting the framework's capacity to uncover organizational principles within chemical latent space.

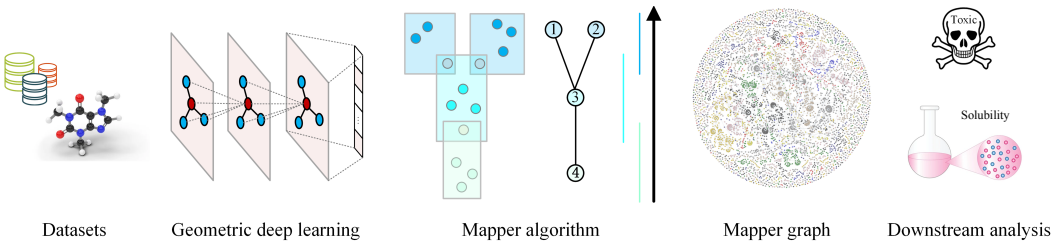


Figure 4 The workflow of the AI-enhanced *Chemical Mapper* for chemical space analysis. From left to right: input of molecular structures; extraction of molecular representations using geometric deep learning models; application of the mapper algorithm to the resulting chemical latent space to construct a mapper graph; and downstream analysis of chemical structures and properties (such as solubility and toxicity) informed by the topology of the mapper graph.

We analyze a large-scale dataset of two million molecules spanning ten scaffolds from the PubChem database [22]. Each molecule is embedded into a 32-dimensional latent space using a pre-trained GEM model. Figure 5 shows the mapper graph constructed from these representations using 100 intervals with 50% overlap, $\epsilon = 1$, and `minPts` $= 10$.

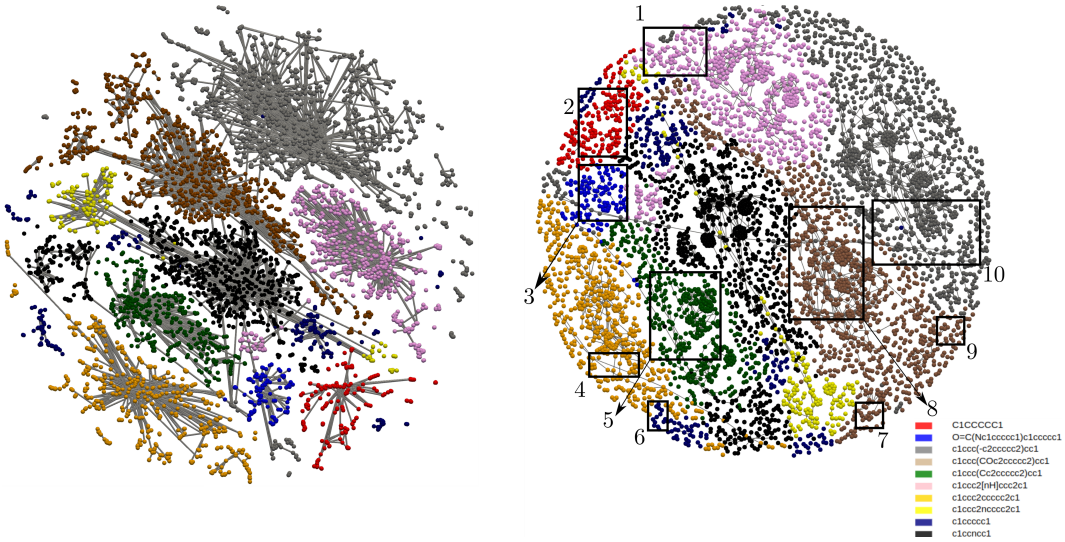


Figure 5 Mapper graph of two million molecular representations from a pre-trained GEM model. The left visualization uses an anchored layout, while the right shows the same graph with a force-directed layout. Numbered regions on the right are examined in detail in subsequent sections.

In Figure 5(left), the mapper graph is drawn using an anchored layout, where each node is initialized at the mean of the 2D t-SNE coordinates of its assigned data points, effectively grounding the graph in the underlying embedding. Figure 5(right) shows the corresponding force-directed layout, which refines this initialization by iteratively adjusting node positions through attractive forces along edges and repulsive forces between nodes to improve separation and reduce visual clutter. For the remainder of the paper, we use the force-directed layout, as it yields clearer and more interpretable visualizations.

We analyze the topological structure of this graph and its relationship to variations in the chemical latent space, focusing on connected components (Section 3.2.1) and branches (Section 3.2.2). We further examine how the graph organization correlates with chemical functions and properties such as toxicity and solubility (Section 3.2.3).

3.2.1 Exploring Connected Components in Chemical Latent Space

As illustrated in Figure 5, the mapper graph consists of multiple connected components, each corresponding to a group of molecules sharing a common scaffold in the latent space. Within a given scaffold, molecules further partition into components of varying sizes, revealing finer-grained structural distinctions.

Figure 6 highlights four connected components, each containing representative molecules drawn from a shared scaffold but distinguished by their functional groups. The top-left component is enriched with molecules featuring amide bonds (O=C–NH–), whereas the bottom-left component consists exclusively of molecules containing sulfate groups (O=S=O). On the right, the top component contains molecules that consistently feature a combination of carboxylic acid (O=C–OH), epoxide (–C–O–C–), and halide substituents. The bottom-right component similarly comprises molecules that simultaneously contain nitro groups (O–N(=O)–O), carboxylic acid, and halide substitutions. This demonstrates that a component can be characterized by a set of functional groups rather than a single one. Together, these topological patterns indicate that the GEM model has learned not only to separate molecules by their underlying scaffolds but also to organize them hierarchically according to their functional groups.

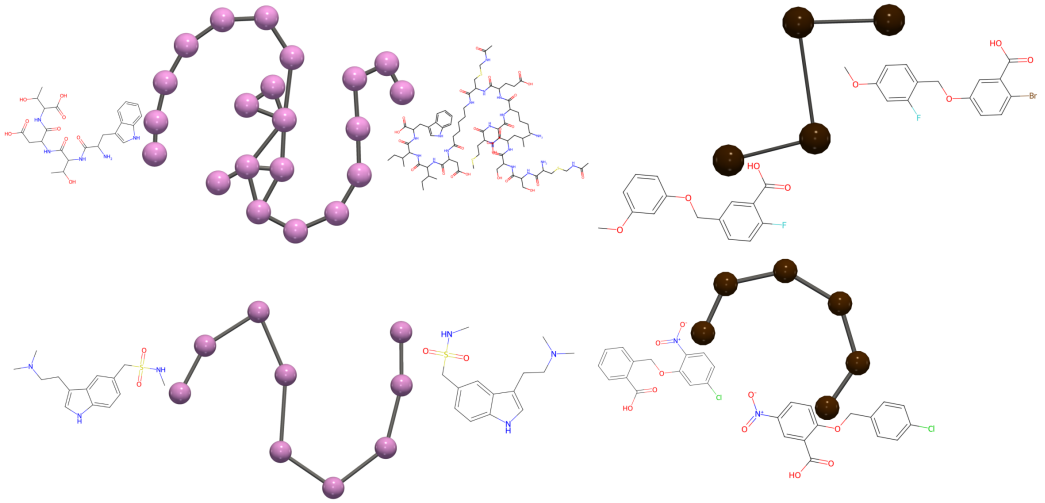


Figure 6 Four chain-like connected components with representative molecules identified by *Chemical Mapper* on the pre-trained GEM model. The components on the left are extracted from region 1 in Figure 5 (right), while the components on the right correspond to regions 9 (top) and 7 (bottom), respectively.

3.2.2 Exploring Chains and Branches in Chemical Latent Space

A *chain* in a mapper graph refers to a linear arrangement of nodes forming a sequential, chain-like structure, whereas *branches* are chains that emerge from a common source node. We investigate the evolution of chemical structures across mapper edges in the latent space by analyzing these chains and branches in the mapper graph.

The components shown in Figure 6 exhibit chain-like topologies, each consisting of a nearly linear sequence of nodes. In the top-left component, the number of amide bonds (O=C–NH–) within the molecules gradually increases from one end of the chain to the other. The GEM model not only groups molecules with a dominant presence of amide bonds into the same component but also differentiates them according to the number of such bonds. This represents a specific instance of a broader pattern in the chemical space, where similar trends are observed across multiple regions of the mapper graph.

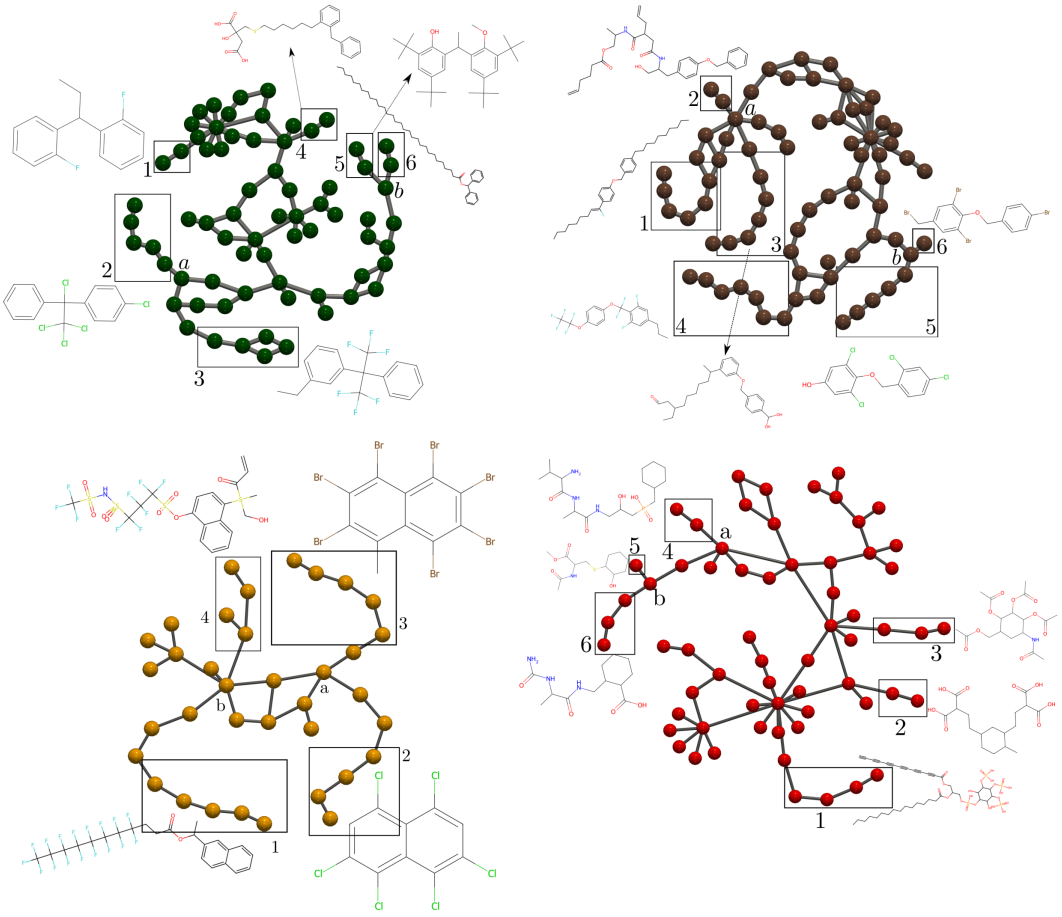


Figure 7 Four connected components with prominent branching structures identified by *Chemical Mapper* on the GEM model. The components are extracted from region 5 (top-left), region 8 (right), region 4 (bottom-left), and region 2 (bottom-right) in Figure 5 (right). Branches are highlighted using numbered boxes, and representative molecules are displayed adjacent to their corresponding branches or indicated with arrows.

In contrast, branches in the mapper graph represent bifurcations that correspond to distinct functional groups. As shown in Figure 7, the numbered branches (highlighted by boxes) reveal fine-scale structural variations among molecular clusters. The top-left component features multiple branches that share the same scaffold (c1ccc(Cc2ccccc2)cc1) but differ in their functional groups. Specifically, branch 1 contains molecules with fluoride groups (–F), whereas branch 4 includes molecules with carboxylic groups (COOH). Branches 5 and 6 emerge from the branching node b, with branch 5 comprising molecules rich in quaternary carbon groups and branch 6 containing molecules with long carbon chains.

These bifurcations not only distinguish functional groups with markedly different chemical structures but also separate those with closely related frameworks that differ only by atomic composition. For instance, from the branching node a, branch 2 primarily contains molecules with chloride (–Cl) and trichloride ($–CCl_3$) groups, whereas branch 3 consists of molecules featuring trifluoride ($–CF_3$) groups.

Similarly, the top-right component in Figure 7 displays multiple branches that reflect comparable bifurcation patterns of functional groups. Branches 1, 2, and 3 originate from node a, containing molecules characterized by long carbon chains, amides (O=C–NH–), and alcohols (–OH), respectively. Branches 5 and 6 bifurcate from node b, with branch 5 including molecules containing chlorides (–Cl) and branch 6 those containing bromides (–Br). Branch 4 comprises molecules with a prominent presence of fluoride (e.g., –F, $–CF_5$) groups.

Similarly, the bottom-left component of Figure 7 is composed of molecules with a high halogen content, and it is notable that all halogen-bearing molecules cluster together within a single component. Within this component, the mapper graph indicates a hierarchical separation of molecules according to the specific halogen substituent. Branch 1 consists of molecules featuring fluorine (–F), branch 2 contains molecules with chloride (–Cl), and branch 3 includes those with bromine (–Br). Notably, branches 2 and 3 both emerge from node a. Although they differ in the halogen substituent, their molecules remain structurally similar because the halogens are directly attached to the underlying aromatic scaffold. Branch 4 comprises molecules that contain both fluorine (–F) and sulfone (O=S=O) groups. Branches 1 and 4 originate from node b, reflecting their shared fluorine-containing molecules.

A similar pattern of organization appears in the bottom-right component of Figure 7. Branches 1, 2, and 3 contain molecules enriched in phosphate (PO_4) and long carbon chains, carboxylic acid (O=C–OH), and aldehyde (O=CH) functional groups, respectively. In contrast, branches 4, 5, and 6 all contain amide functional groups (O=C–NH–). Rather than forming a single branch, these three branches diverge hierarchically, reflecting additional chemical variations among the molecules. Branch 4 diverges first at node a, indicating the presence of a prominent phosphoryl functional group (PO_2), in addition to the amide group. From the remaining branch emerging from node a, a further split occurs at node b. This subsequent bifurcation reflects an additional layer of chemical variation: branch 5 contains molecules featuring sulfur substituents (–S–), whereas branch 6 includes molecules enriched in carboxylic acid functional groups (O=C–OH). This hierarchical branching structure shows how the mapper graph progressively separates molecules based on both their dominant and secondary functional groups.

3.2.3 Mapper for Chemical Function Analysis

Given the strong relationship between molecular structure and chemical function, mapper graphs provide a powerful framework for interpreting chemical properties. We use the water solubility and toxicity datasets from Deep-PK [31], comprising 9,964 molecules with solubility labels and 10,130 molecules with toxicity labels. Both properties are reported on a logarithmic scale. Using these datasets, we train the GEM model [13] to predict properties for the two million molecules analyzed in this study. Specifically, 90% of the labeled data are used for training and the remaining 10% for validation, with model parameters optimized based on validation performance. Once the GEM model reaches its optimal accuracy, it is employed to predict water solubility and toxicity for unlabeled molecules, thereby assigning property labels to previously uncharacterized chemical structures for downstream analysis.

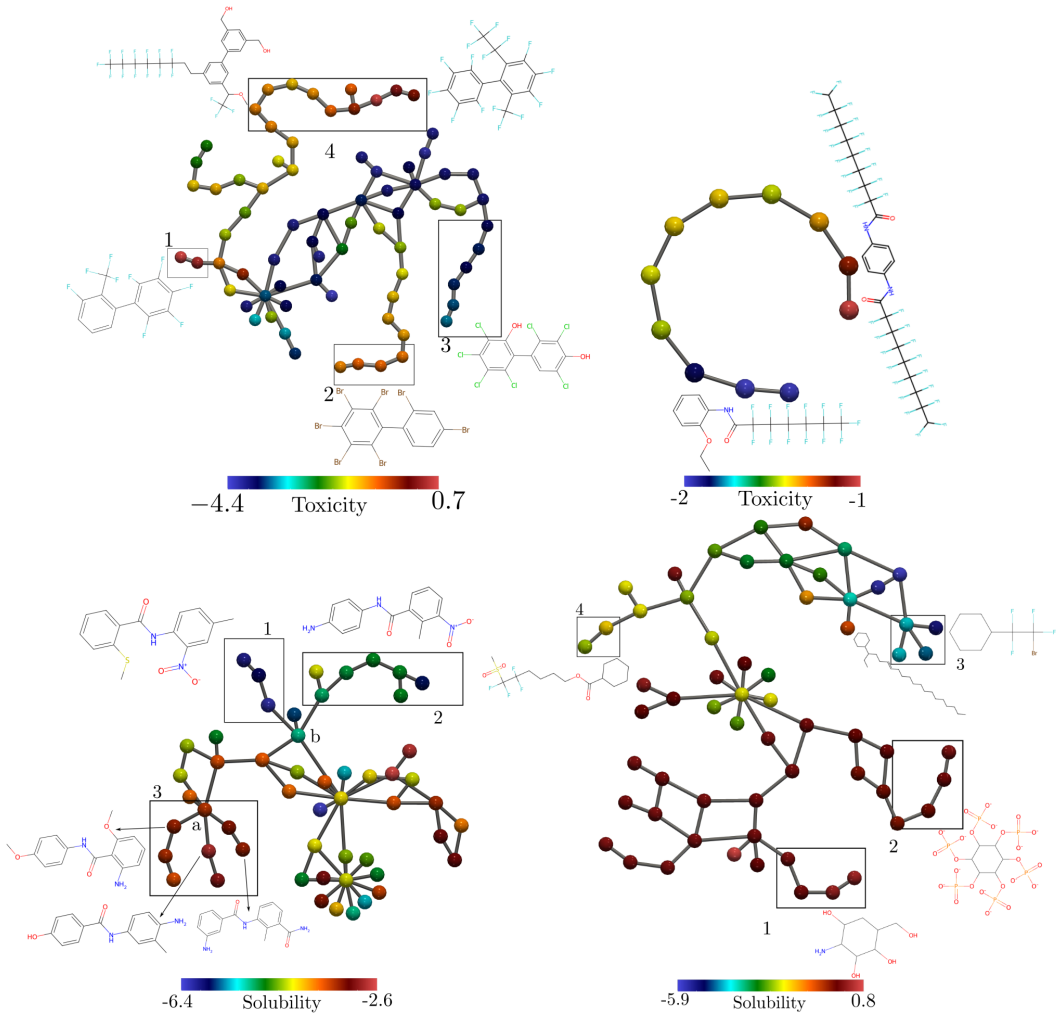


Figure 8 Four connected components with prominent branching structures or chains identified by *Chemical Mapper* and colored according to averaged chemical properties. The components are extracted from region 10 (top-left), region 6 (top-right), region 3 (bottom-left), and region 2 (bottom-right) in Figure 5 (right). Branches are highlighted with numbered boxes, and representative molecules are displayed adjacent to their corresponding branches or indicated with arrows.

Our key finding is that the mapper graph effectively visualizes the organization of molecules with similar chemical properties into cohesive regions, while molecules spanning different property ranges are distributed across distinct branches. This organization aligns closely with structural characteristics: the branching patterns in the mapper graph reflect variations in functional groups, the primary determinants of chemical behavior.

Figure 8 illustrates four connected components from the mapper graph in Figure 5. In the top panel, nodes are colored by molecular toxicity, and in the bottom, by water solubility; colormaps are adjusted for each component to better highlight diversity. In both cases, nodes within each branch share similar colors or exhibit gradual color transitions, indicating smooth variation in the corresponding property. In Figure 8 (top-left), branch 1 contains highly toxic molecules, likely due to the strong presence of fluoride groups (–F). Branch 2, which shows higher toxicity than branch 3, is enriched with bromide-containing (–Br) molecules, while

branch 3 is characterized by chloride (–Cl) substitution. Branch 4 is particularly notable: toxicity increases progressively along the branch. Although all molecules in this branch contain fluoride groups, the less toxic end includes additional functional groups, whereas the highly toxic end consists solely of fluoride-bearing (–F) molecules. Representative molecular structures are displayed alongside each branch.

In Figure 8 (top-right), the component forms a single linear chain in which toxicity increases steadily along its length. This gradual rise in toxicity parallels the chemical trend observed in the structures: all molecules in the chain contain fluorine substituents (–F), but the number of fluorine functional groups increases progressively from one end of the chain to the other. As a result, molecules at the high-toxicity end exhibit a substantially heavier fluorine presence compared to those at the low-toxicity end.

A similar pattern is observed in the component in Figure 8 (bottom-right). Within each branch, nodes display uniform or gradually changing colors. Branch 1 consists of highly soluble molecules, primarily due to the presence of alcohol (–OH) and amine (–NH_2) groups. Branch 2 also contains highly soluble molecules, likely attributable to phosphate (–PO_4) groups. This is interpretable, as phosphate-containing molecules (–PO_4) appear in high-solubility regions, due to their charged and polar nature. These functional groups correspond to identifiable subgraphs (C–O–H and P–O–O patterns) that the GEM model learns to encode, forming distinct branches in the mapper graph. In Area 3, the branches are dominated by molecules with halide substitutions (e.g., –Br, –F) or very long carbon chains (–$(CH_2)_n$–), which generally correspond to low water solubility. Branch 4 contains molecules of moderate solubility, as they appear to contain sulfone (–SO_2–) groups that increase solubility, but also halides that reduce it, resulting in an intermediate solubility profile.

A comparable pattern emerges in Figure 8 (bottom-left) component, where each branch contains molecules with either uniform or smoothly varying solubility levels. Branch 1 consists of molecules with low solubility, primarily due to the presence of nitro groups (–NO_2). Branch 2, which also contains nitro-bearing (–NO_2) molecules, exhibits moderately higher solubility because these molecules additionally feature amine functional groups (–NH_2), which enhance aqueous solubility. Note, branches 1 and 2 diverge from node b, reflecting their shared nitro substitution. Area 3 contains three branches, each composed of molecules with high solubility. These molecules predominantly feature epoxide (–C–O–C–), alcohol (–OH), or amide (O=C–NH–) functional groups. All three branches emerge from node a, consistent with the fact that their molecular structures are closely related, differing mainly in the specific solubility-enhancing functional group present.

Finally, for completeness, we present the full mapper graph of the chemical latent space from the pre-trained GEM model, where nodes are colored according to predicted toxicity (left) and solubility (right) in Figure 9.

3.3 Exploring Chemical Latent Spaces from Different Learning Models

In this section, we demonstrate how mapper graphs can serve as a powerful tool for characterizing and evaluating molecular representations learned by geometric deep learning models such as GraphMAE and Mole-BERT. We further show that mapper graphs enable qualitative comparisons of the learning capabilities between pre-trained and fine-tuned models.

Following the procedure for the GEM model (Section 3.2), we apply *Chemical Mapper* to the molecular representations of the same two million molecules from the PubChem database [22], learned by the pre-trained GraphMAE model. A subgraph of the resulting mapper graph is shown in Figure 10 (left), where each node is represented as a pie chart

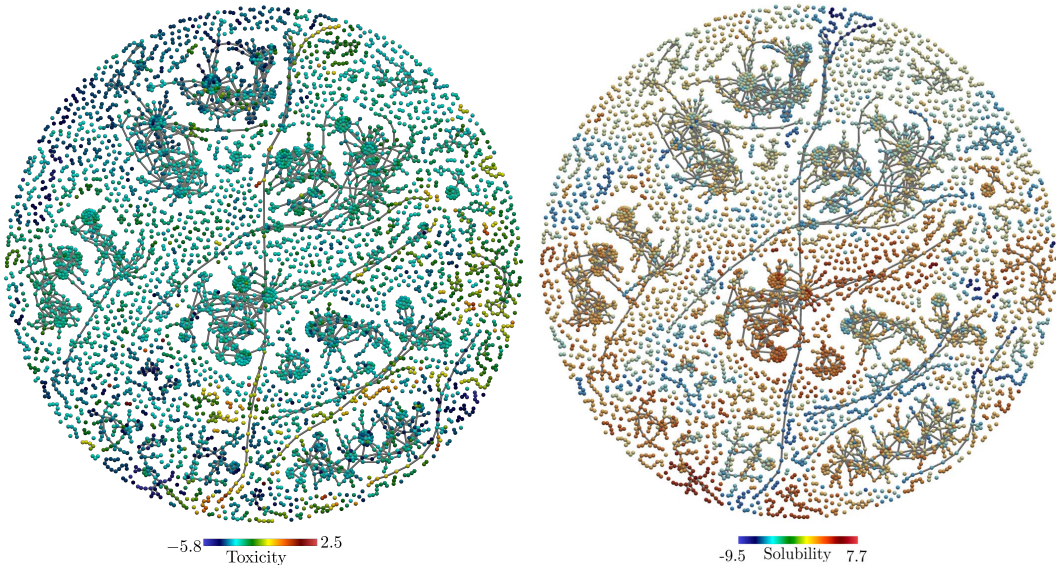


Figure 9 Mapper graphs of molecular representations from the pre-trained GEM model, computed using the same experimental settings as in Figure 5. Nodes are colored according to molecular toxicity (left) and solubility (right), as discussed in Section 3.2.3 and Section 2.

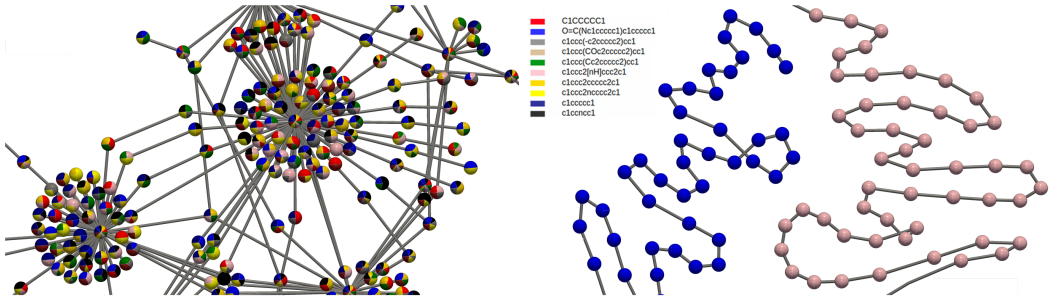


Figure 10 Partial views of mapper graphs derived from latent representations of the pre-trained (left) and fine-tuned (right) GraphMAE models. Fine-tuning refines and restructures the latent space, revealing its evolution during training. Both graphs are computed with 500 intervals, 40% overlap, and DBSCAN parameters $\epsilon = 5$ and `minPts` $= 5$.

indicating the distribution of molecular scaffolds within that node. Each node clearly contains a mixture of scaffolds, suggesting that the pre-trained GraphMAE model does not separate the chemical space according to scaffold structure (cf. Section 3.2.1). Moreover, the mapper graph exhibits no clear branching patterns, in contrast to the structured branches observed for the GEM model (Section 3.2.2).

Subsequently, the pre-trained GraphMAE model is fine-tuned on the task of predicting molecular scaffolds, and representations for the same two million molecules are extracted from the fine-tuned model. Using identical parameters, a new mapper graph is constructed. Figure 10 (right) shows a portion of this graph, where the nodes are noticeably more homogeneous, with molecules sharing the same scaffold clustering together. Fine-tuning the GraphMAE model, therefore, induces a clearer partitioning of the chemical space according to molecular scaffolds. The resulting mapper graph contains 1,606 nodes – a substantial

increase from the 420 nodes in the mapper graph of the pre-trained model – indicating that fine-tuning expands the representational space and more effectively separates molecules with different scaffolds.

Furthermore, the mapper graph derived from the fine-tuned model consists predominantly of chains and exhibits few branches. As discussed in Section 3.2.2, branching in the chemical space reflects distinctions among functional groups. Because the fine-tuning task focuses solely on scaffold prediction and does not consider functional groups, the lack of branching suggests that the chemical space is organized primarily around scaffold structure rather than functional group variation.

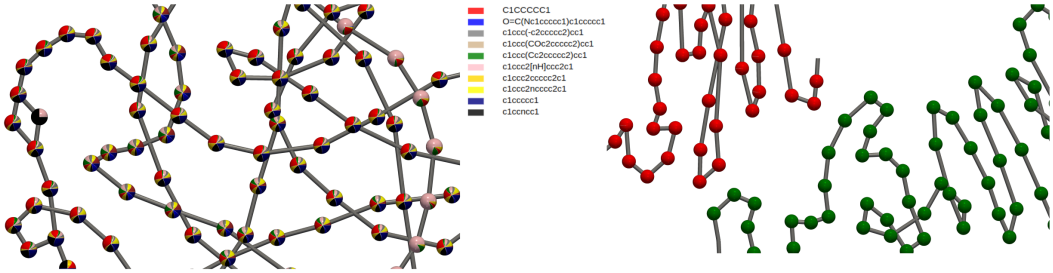


Figure 11 Partial views of mapper graphs of molecular representations from the pre-trained (left) and fine-tuned (right) Mole-BERT model. Nodes are colored by scaffolds.

We conduct a similar experiment with Mole-BERT on the same two million PubChem molecules, obtaining results qualitatively consistent with GraphMAE. Figure 11 compares mapper graphs from the pre-trained (left) and fine-tuned (right) Mole-BERT models for the scaffold classification task.

For completeness, we may also compare the pre-trained and fine-tuned GEM model. Figure 12 shows the complete mapper graph of the chemical latent space derived from the GEM model fine-tuned on the scaffold classification task, with nodes colored by molecular scaffolds. We perform a comparative analysis between the pre-trained and fine-tuned GEM models. As shown in Figure 5 and Figure 12, the mapper graph of the fine-tuned GEM model exhibits the same qualitative trends observed in the pre-trained model (Section 3.2): molecules with different scaffolds remain clearly separated into distinct connected components, and branching structures continue to reflect variations in functional groups.

Because the chemical space produced by the pre-trained GEM model already partitions molecules according to scaffolds (as shown in Figure 5), fine-tuning on this task does not substantially affect the purity of the cluster nodes. However, comparing the mapper graphs in Figure 5 and Figure 12 reveals notable topological changes. Certain scaffold classes become more dispersed, exhibiting a larger number of isolated nodes and fewer interconnections (e.g., c1ccccc1), whereas others contract significantly (e.g., c1ccc(-c2ccccc2)cc1 and c1ccc(COc2ccccc2)cc1). Investigating the causes and implications of these topological changes represents an important direction for future research.

4 Conclusion, Discussion and Future Work

In conclusion, the immense size and complexity of chemical space present substantial challenges for analysis and discovery. In this work, we introduce *Chemical Mapper*, a mapper framework that enables effective visualization, search, exploration, and interpretation of chemical latent spaces. Our approach offers several key advantages. First, similar to tradi-

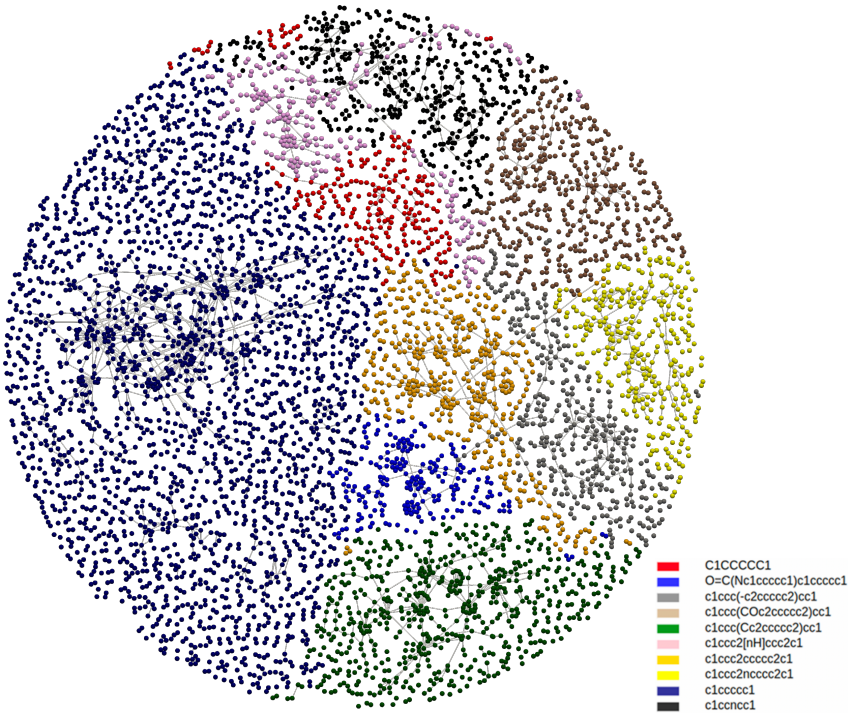


Figure 12 Mapper graph of molecular representations from the fine-tuned GEM model, colored by molecular scaffolds.

tional clustering methods, the mapper graph organizes molecules with similar properties into coherent groups. Second, beyond simple clustering, the graph's topological structures, such as connected components and branches, capture the continuous variations and evolution among molecular structures. Third, by jointly encoding both structural clustering and topological organization, mapper graphs provide a powerful lens for understanding chemical functions and properties. Finally, *Chemical Mapper* serves not only as an analysis tool but also as a diagnostic framework for evaluating chemical representations: it allows qualitative assessment of how well deep learning models capture meaningful chemical patterns.

Our framework provides a principled and interpretable approach for navigating chemical latent space, revealing its intrinsic organization and supporting both the development and evaluation of molecular deep learning methods. We conclude with a discussion of parameter tuning, stability, extensibility, and opportunities for enhanced human–AI collaboration.

4.1 Parameter Tuning

The *Chemical Mapper* framework involves several key parameters: the number of intervals m, the percentage of overlap p, and the DBSCAN clustering parameters, including the neighborhood radius ϵ and the minimum number of points `minPts` required to form a core point. These parameters are collectively denoted as $(m, p, \epsilon, \texttt{minPts})$. In practice, they are often hand-tuned (as in this study), with best practice being to select values within ranges where the mapper graph structures (such as branches, loops, and connected components) remain stable. Prior works have investigated automated parameter selection for mapper graphs [1, 6, 7, 8], and ϵ can be estimated using the elbow method [12, 53].

4.2 Stability

Theoretical studies have investigated the structural stability of mapper graphs [4, 7] and their multiscale variants [11]. In particular, mapper graphs are stable under perturbations of the data $(\mathbb{X}, f)$, with stability depending on how the cover $\mathcal{U}$ aligns with the critical values of f [4]. However, these theoretical guarantees rely on statistical assumptions that are often not satisfied in real-world datasets. Motivated by this gap, we provide empirical evidence of stability in Figure 13.

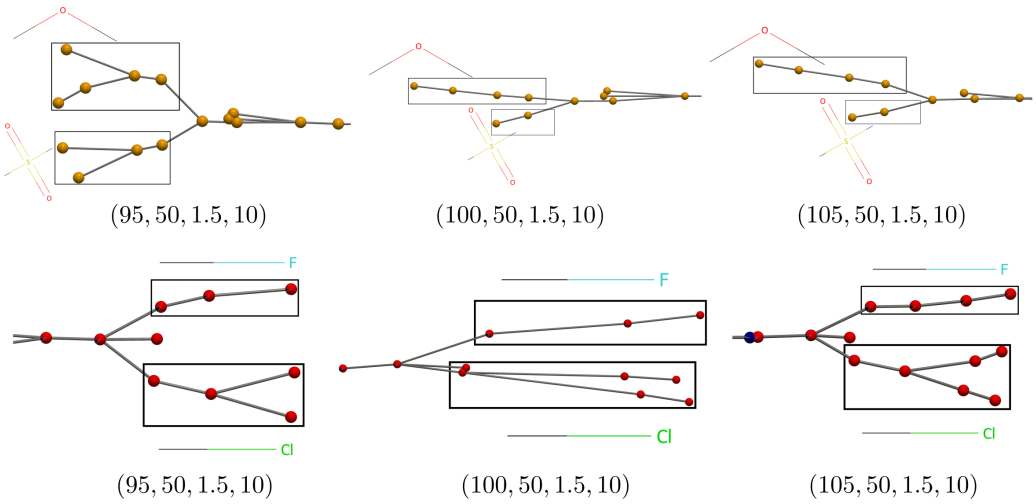


Figure 13 Two examples of local branches in mapper graphs generated under varying parameter settings $(m, p, \epsilon, \texttt{minPts})$, where p, ϵ, and `minPts` are fixed and m varies from 95 to 105. In both top and bottom cases, a component with the same branching structure and the same set of molecules persists across different m values, illustrating the stability of the mapper construction.

Building on these findings, we examine the stability of mapper graphs under varying parameter configurations using two million molecular representations from PubChem, generated by the pre-trained GEM model (same as the dataset used in Section 3). As shown in Figure 13, local branches in the mapper graphs remain consistent across suitable ranges of m, where each branch corresponds to molecules sharing a functional group, indicated alongside the branch. In particular, the first row in Figure 13 highlights connected components containing molecules with –O– and –(O=S=O)– functional groups, exhibiting the same bifurcation pattern across three values of m. Similarly, the second row in Figure 13 shows a bifurcation between molecules containing –F and –Cl groups across different interval numbers.

Overall, this experiment provides an initial demonstration of the robustness of the mapper framework in capturing chemically meaningful topological structures across parameter settings. Although parameter variations affect the granularity of branching, the key separations corresponding to distinct functional groups remain stable. Future work should investigate how chemical semantics relate to parameter choices, guiding principled parameter selection and further enhancing the mapper's interpretability and applicability in cheminformatics.

4.3 Extendability and Human-AI Collaboration

We plan to extend *Chemical Mapper* to study structural motifs beyond scaffolds, including functional groups as well as aromatic and reactivity motifs. The recent Explainable Mapper framework [49] provides an effective strategy for analyzing high-dimensional latent spaces

by integrating topological representations with agentic large language model (LLM)–based explanations. In the chemical domain – where latent spaces may encode relationships among molecular structure, reactivity, and function – this approach opens up rich opportunities for human–AI collaboration. In such a setting, rather than relying solely on human experts to explain mapper elements (Section 3.2), agents powered by LLMs or VLMs (Vision–Language Models) can propose distinguishing descriptors (e.g., fingerprints, substructures, or property ranges) and generate hypotheses about the chemical attributes that unify a given mapper element – such as the connected components, chains, and branches described in Section 3.2. Human experts can then validate or refine these hypotheses using their domain knowledge. We leave this direction for future work.

References

1 Enrique Alvarado, Robin Belton, Emily Fischer, Kang-Ju Lee, Sourabh Palande, Sarah Percival, and Emilie Purvine. G-Mapper: Learning a cover in the mapper construction. *SIAM Journal on Mathematics of Data Science*, 7(2):572–596, 2025.

2 Silvia Biasotti, Daniela Giorgi, Michela Spagnuolo, and Bianca Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1-3):5–22, 2008. doi:10.1016/J.TCS.2007.10.018.

3 Maria Boulougouri, Pierre Vandergheynst, and Daniel Probst. Molecular set representation learning. *Nature Machine Intelligence*, 6(7):754–763, 2024. doi:10.1038/S42256-024-00856-0.

4 Adam Brown, Omer Bobrowski, Elizabeth Munch, and Bei Wang. Probabilistic convergence and stability of random mapper graphs. *Journal of Applied and Computational Topology*, 5(1):99–140, 2021. doi:10.1007/S41468-020-00063-X.

5 Rickard Brüel Gabrielsson and Gunnar Carlsson. Exposition and interpretation of the topology of neural networks. In *18th IEEE International Conference On Machine Learning And Applications*, pages 1069–1076, 2019. doi:10.1109/ICMLA.2019.00180.

6 Quang-Thinh Bui, Bay Vo, Hoang-Anh Nguyen Do, Nguyen Quoc Viet Hung, and Vaclav Snasel. F-Mapper: A fuzzy mapper clustering algorithm. *Knowledge-Based Systems*, 189(C), 2020.

7 Mathieu Carriere and Steve Oudot. Structure and stability of the one-dimensional Mapper. *Foundations of Computational Mathematics*, 18:1333–1396, 2018. doi:10.1007/S10208-017-9370-Z.

8 Nithin Chalapathi, Youjia Zhou, and Bei Wang. Adaptive covers for mapper graphs using information criteria. In *IEEE International Conference on Big Data*, pages 3789–3800, 2021. doi:10.1109/BIGDATA52589.2021.9671324.

9 Lixue Cheng, Jiace Sun, and Thomas F. III Miller. Accurate molecular-orbital-based machine learning energies via unsupervised clustering of chemical space. *Journal of Chemical Theory and Computation*, 18(8):4826–4835, 2022.

10 Connor W Coley. Defining and exploring chemical spaces. *Trends in Chemistry*, 3(2):133–145, 2021.

11 Tamal K. Dey, Facundo Mémoli, and Yusu Wang. Multiscale mapper: Topological summarization via codomain covers. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 997–1013, 2016. doi:10.1137/1.9781611974331.CH71.

12 Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996. URL: http://www.aaai.org/Library/KDD/1996/kdd96-037.php.

13 Xiaomin Fang, Lihang Liu, Jieqiong Lei, Donglong He, Shanzhuo Zhang, Jingbo Zhou, Fan Wang, Hua Wu, and Haifeng Wang. Geometry-enhanced molecular representation learning for property prediction. *Nature Machine Intelligence*, 4(2):127–134, 2022. doi:10.1038/S42256-021-00438-4.

14 Andrej Gisbrecht and Barbara Hammer. Data visualization by nonlinear dimensionality reduction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(2):51–73, 2015. doi:10.1002/WIDM.1147.

15 Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

16 Hamid Hadipour, Chengyou Liu, Rebecca Davis, Silvia T. Cardona, and Pingzhao Hu. Deep clustering of small molecules at large-scale via variational autoencoder embedding and k-means. *BMC Bioinformatics*, 23(4):132, 2022. doi:10.1186/S12859-022-04667-1.

17 Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. GraphMAE: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2022. doi:10.1145/3534678.3539321.

18 Anil K Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010. doi:10.1016/J.PATREC.2009.09.011.

19 Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999. doi:10.1145/331499.331504.

20 Antonius P. A. Janssen, Sebastian H. Grimm, Ruud H. M. Wijdeven, Eelke B. Lenselink, Jacques Neefjes, Constant A. A. van Boeckel, Gerard J. P. van Westen, and Mario van der Stelt. Drug Discovery Maps, a machine learning model that visualizes and predicts kinome–inhibitor interaction landscapes. *Journal of Chemical Information and Modeling*, 59(3):1221–1229, 2019. doi:10.1021/ACS.JCIM.8B00640.

21 Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. PubChem 2023 update. *Nucleic Acids Research*, 51(D1):D1373–D1380, 2023.

22 Sunghwan Kim, Paul A Thiessen, Evan E Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A Shoemaker, et al. PubChem substance and compound databases. *Nucleic Acids Research*, 44(D1):D1202–D1213, 2016.

23 John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

24 Christopher Lipinski and Andrew Hopkins. Navigating chemical space for biology and medicine. *Nature*, 432(7019):855–861, 2004.

25 Yu Liu, Cole Mathis, Michał Dariusz Bajczyk, Stuart M. Marshall, Liam Wilbraham, and Leroy Cronin. Exploring and mapping chemical space with molecular assembly trees. *Science Advances*, 7(39):eabj2465, 2021.

26 Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. doi:10.1109/TIT.1982.1056489.

27 Gerald M. Maggiora and Jürgen Bajorath. Chemical space networks: a powerful new paradigm for the description of chemical space. *Journal of Computer-Aided Molecular Design*, 28:795–802, 2014. doi:10.1007/S10822-014-9760-0.

28 Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi:10.21105/JOSS.00861.

29 Oscar Méndez-Lucio, Mazen Ahmad, Ehecatl Antonio del Rio-Chanona, and Jörg Kurt Wegner. A geometric deep learning approach to predict binding conformations of bioactive molecules. *Nature Machine Intelligence*, 3(12):1033–1039, 2021. doi:10.1038/S42256-021-00409-9.

30 Elizabeth Munch and Bei Wang. Convergence between categorical representations of Reeb space and mapper. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 53:1–53:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SOCG.2016.53.

31 Yoochan Myung, Alex G.C. de Sá, and David B. Ascher. Deep-PK: deep learning for small molecule pharmacokinetic and toxicity prediction. *Nucleic Acids Research*, page gkae254, 2024.

32 Toshiki Ochiai, Tensei Inukai, Manato Akiyama, Kairi Furui, Masahito Ohue, Nobuaki Matsumori, Shinsuke Inuki, Motonari Uesugi, Toshiaki Sunazuka, Kazuya Kikuchi, et al. Variational autoencoder-based chemical latent space for large molecular structures with 3D complexity. *Communications Chemistry*, 6(1):249, 2023.

33 Gbeminiyi John Oyewole and George Alex Thopil. Data clustering: application and trends. *Artificial Intelligence Review*, 56(7):6439–6475, 2023. doi:10.1007/S10462-022-10325-Y.

34 Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

35 Kenneth López Pérez, Vicky Jung, Lexin Chen, Kate Huddleston, and Ramón Alain Miranda-Quintana. BitBIRCH: efficient clustering of large molecular libraries. *Digital Discovery*, 4:1042–1051, 2025.

36 Daniel Probst and Jean-Louis Reymond. Visualization of very large high-dimensional data sets as minimum spanning trees. *Journal of Cheminformatics*, 12(1):12, 2020. doi:10.1186/S13321-020-0416-X.

37 Emilie Purvine, Davis Brown, Brett Jefferson, Cliff Joslyn, Brenda Praggastis, Archit Rathore, Madelyn Shapiro, Bei Wang, and Youjia Zhou. Experimental observations of the topology of convolutional neural network activations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):9470–9479, 2023. doi:10.1609/AAAI.V37I8.26134.

38 Archit Rathore, Nithin Chalapathi, Sourabh Palande, and Bei Wang. TopoAct: Visually exploring the shape of activations in deep learning. *Computer Graphics Forum*, 40(1):382–397, 2021. doi:10.1111/CGF.14195.

39 Archit Rathore, Yichu Zhou, Vivek Srikumar, and Bei Wang. TopoBERT: Exploring the topology of fine-tuned word representations. *Information Visualization*, 22(3):186–208, 2023. doi:10.1177/14738716231168671.

40 Georges Reeb. Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique [on the singular points of a completely integrable pfaff form or of a numerical function]. *Comptes Rendus Acad. Sciences Paris*, 222:847–849, 1946.

41 Jean-Louis Reymond. The chemical space project. *Accounts of Chemical Research*, 48(3):722–730, 2015.

42 Jean-Louis Reymond, Ruud van Deursen, Lorenz C. Blum, and Lars Ruddigkeit. Chemical space as a source for new drugs. *Medicinal Chemistry Communications*, 1:30–38, 2010.

43 Vincent F Scalfani, Vishank D Patel, and Avery M Fernandez. Visualizing chemical space networks with RDKit and NetworkX. *Journal of Cheminformatics*, 14(1):87, 2022. doi:10.1186/S13321-022-00664-X.

44 Cong Shen, Jiawei Luo, and Kelin Xia. Molecular geometric deep learning. *Cell Reports Methods*, 3(11), 2023.

45 Gurjeet Singh, Facundo Mémoli, and Gunnar E. Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. *Eurographics Symposium on Point-Based Graphics*, 2:091–100, 2007. doi:10.2312/SPBG/SPBG07/091-100.

46 Murat Cihan Sorkun, Dajt Mullaj, J. M. Vianney A. Koelman, and Süleyman Er. ChemPlot, a Python library for chemical space visualization. *Chemistry-Methods*, 2(7):e202200005, 2022.

47 Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

48 Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. Mole-BERT: Rethinking pre-training graph neural networks for molecules. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

49 Xinyuan Yan, Rita Sevastjanova, Sinie van der Ben, Mennatallah El-Assady, and Bei Wang. Explainable Mapper: Charting LLM embedding spaces using perturbation-based explanation and verification agents. arXiv preprint arXiv:2507.18607, 2025.

50 Shuwen Yang, Ziyao Li, Guojie Song, and Lingsheng Cai. Deep molecular representation learning via fusing physical and chemical information. *Advances in Neural Information Processing Systems*, 34:16346–16357, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/884d247c6f65a96a7da4d1105d584ddd-Abstract.html.

51 Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997. doi:10.1023/A:1009783824328.

52 Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-Mol: A universal 3D molecular representation learning framework. In *Proceedings of the 11th International Conference on Learning Representations*. International Conference on Learning Representations, 2023.

53 Youjia Zhou, Nithin Chalapathi, Archit Rathore, Yaodong Zhao, and Bei Wang. Mapper Interactive: A scalable, extendable, and interactive toolbox for the visual exploration of high-dimensional data. In *IEEE 14th Pacific Visualization Symposium*, pages 101–110, 2021. doi:10.1109/PACIFICVIS52677.2021.00021.

54 Youjia Zhou, Helen Jenne, Davis Brown, Madelyn Shapiro, Brett Jefferson, Cliff Joslyn, Gregory Henselman-Petrusek, Brenda Praggastis, Emilie Purvine, and Bei Wang. Comparing mapper graphs of artificial neuron activations. In *Topological Data Analysis and Visualization*, pages 41–50, 2023.

55 Youjia Zhou, Yi Zhou, Jie Ding, and Bei Wang. Visualizing and analyzing the topology of neuron activations in deep adversarial training. In Timothy Doster, Tegan Emerson, Henry Kvinge, Nina Miolane, Mathilde Papillon, Bastian Rieck, and Sophia Sanborn, editors, *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning*, volume 221 of *Proceedings of Machine Learning Research*, pages 134–145. PMLR, 2023. URL: https://proceedings.mlr.press/v221/zhou23a.html.

D-GRIL: End-To-End Topological Learning with 2-Parameter Persistence

Soham Mukherjee[12]
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Shreyas N. Samaga[1]
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Cheng Xin
Department of Computer Science, Rutgers University, New Brunswick, NJ, USA

Steve Oudot
GeomeriX team, Inria Saclay and École polytechnique, Palaiseau, France

Tamal K. Dey
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Abstract

End-to-end topological learning using 1-parameter persistence is well-known. We show that the framework can be enhanced using 2-parameter persistence by adopting a recently introduced 2-parameter persistence based vectorization technique called GRIL. We establish a theory for gradient *descent* on GRIL producing D-GRIL. We show that D-GRIL can be used to learn a bifiltration function on benchmark graph datasets. Further, we exhibit that this framework can be applied in the context of bio-activity prediction in drug discovery.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Computing methodologies → Machine learning

Keywords and phrases Topological Data Analysis, Persistent Homology, Multiparameter Persistence, Graph Learning, Graph Neural Networks

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.79

Related Version *Full Version*: `https://arxiv.org/pdf/2406.07100` [40]

Supplementary Material *Software*: `https://github.com/TDA-Jyamiti/d-gril/` [39]
archived at `swh:1:dir:379b8266de5d11b54b8041b9a4af9b9dde1fa254`

Funding This work is partially supported by NSF grants CCF 2049010 and DMS 2301360.

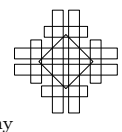
1 Introduction

In recent years, persistent homology, one of the flagship concepts of Topological Data Analysis (TDA), has found its use in many fields such as neuroscience, material science, sensor networks, shape recognition, gene expression data analysis and many more [12, 25]. The performance of machine learning models such as Graph Neural Networks (GNNs) can be improved by augmenting topological information captured by persistent homology [8, 15, 29, 30]. Classical persistent homology, also known as 1-parameter persistence, captures the evolution of topological structures in a simplicial complex K following a filter function $f\colon K \to \mathbb{R}$. The evolution of topological structures, in this case, can be completely characterized and compactly represented as *persistence diagrams* or equivalently *barcodes* [21, 50]. These

[1] Equal contribution
[2] Now in Cadence Design Systems, San José



© Soham Mukherjee, Shreyas N. Samaga, Cheng Xin, Steve Oudot, and Tamal K. Dey;
licensed under Creative Commons License CC-BY 4.0



persistence diagrams or barcodes can be vectorized [1, 2, 8, 28, 31, 42] and used in machine learning pipelines. In most applications, the simplicial complex K is given and the choice of the filter function f depends on the application. Choosing an appropriate filter function can be challenging. To avoid this, in [27], the authors proposed an end-to-end learning framework to learn the filter function rather than relying on making the right choice. They showed that learning the filter function performs better than the standard choices of filter functions on many graph datasets.

To obtain richer topological information, we can consider an $\mathbb{R}^n$-valued filter function instead of a scalar filter function. This leads to the structure of a multiparameter persistence module instead of 1-parameter persistence module. Multiparameter persistence modules, unlike their 1-parameter counterparts, can *not* be classified completely using a discrete invariant [5]. Consequently, vectorizing multiparameter persistence modules while retaining as much topological information as possible has become a challenging problem. There are many recent works in this direction which use incomplete invariants such as *rank invariant* [5] or equivalently *fibered barcodes* [33], multiparameter persistence images [6], multiparameter persistence landscapes [45], multiparameter persistence kernel [11], vectorization of signed barcodes [35], topological fingerprinting for virtual screening using multidimensional persistence [16], effective multidimensional persistence [10] or *generalized rank invariant* [48]. The authors, in all these works, show that 2-parameter persistence methods perform better than 1-parameter persistence methods on many graph and time-series datasets, suggesting that an $\mathbb{R}^2$-valued filter function, indeed, captures richer topological information than a scalar filter function. However, in all these works, one needs to make a suitable choice for an $\mathbb{R}^2$-valued filter function. Akin to the 1-parameter setup, learning the filter function rather than choosing a filter function can prove to be more informative and beneficial for the task at hand.

In this paper, we propose an end-to-end learning framework using 2-parameter persistent homology based on the vectorization Gril introduced by [48]. We show that Gril is piecewise affine and give an explicit formula for the differential of Gril. The results discussed in [7, 14] help us to show the convergence of stochastic sub-gradient descent on Gril. We use these results to build a differentiable topological layer D-Gril, consequently an end-to-end learning pipeline. We use D-Gril for bio-activity prediction and provide experimental results. One of the key advantages of using TDA in the context of bio-activity prediction is its ability to capture and analyze the shape and structural characteristics of molecules in a manner that traditional methods might overlook. Molecules possess intricate shapes and topological features that directly influence their interactions with biological targets. TDA can uncover these subtleties, allowing for a deeper understanding of structure-activity relationships. We also show that D-Gril can be used, more generally, for bifiltration learning on graphs and provide experimental results where we compare with existing multiparameter persistence methods on benchmark graph datasets.

In a recent work, authors of [44] present an end-to-end topological learning method by developing a general framework suitable for a class of invariants of multiparameter persistence. Our approach though specific to Gril is more direct, uses simpler arguments, and is faster for training.

2 Overview

Our main construct is Gril, introduced in [48] for vectorizing a 2-parameter persistence module. Gril is defined in the same spirit as persistence landscapes [2], introduced for 1-parameter persistence. The landscape uses the rank function over the interval decomposition known for 1-parameter persistence modules. To extend the concept to 2-parameter persistence

modules, one faces two main challenges: (i) in general, persistence modules beyond 1-parameter may not decompose into intervals, (ii) the usual rank function cannot be defined for intervals that are not rectangular. GRIL solves (i) by defining the landscape function over a fixed set of two-dimensional intervals called ℓ-*worms* instead of the support of the indecomposables of the modules, and solves (ii) by considering a generalized version of the rank function termed as *generalized rank* (Definition 3).

An interesting property of the generalized rank over intervals is that it monotonically decreases as the intervals get larger. Thus, given fixed parameters $k > 0$ and $\ell > 0$, the width d of an ℓ-worm can be increased until the generalized rank over the worm drops below k. The maximum value of the width thus obtained, $d_{\mathbf{p}}$, is the value of GRIL at the worm center point $\mathbf{p}$ for k and ℓ.

GRIL yields a vector representation of the 2-parameter persistence module formed from a given simplicial complex and a bifiltration function (formally defined in section 3). However, there is no provision to learn this bifiltration function. To learn a suitable bifiltration function, we build a differentiable layer called D-GRIL. We show in section 6 that the learnt bifiltration function performs better than some of the popular bifiltration functions known in TDA supporting the need for end-to-end learning.

3 Background

We review and recall some definitions and results in this section. Section 3.1 primarily consists of concepts in multiparameter persistent homology. Section 3.3 consists of concepts and results that establish the conditions required for the convergence of stochastic sub-gradient descent. Section 3.2 contains some background which is required to prove that GRIL satisfies these conditions.

3.1 Multiparameter persistent homology

In this subsection, we briefly review the concepts in multiparameter persistent homology. We focus, primarily, on 2-parameter persistent homology. For detailed definitions, refer [20, 22]. We begin by recalling the definition of a simplicial complex. Given a finite vertex set V, a simplicial complex $K = K(V)$ defined on this vertex set is a collection of subsets of V such that if a subset $\sigma \subseteq V$ is in K, then all proper subsets $\tau \subset \sigma$ are also in K. Each element in K with cardinality $k+1$ is called a k-simplex or simply a simplex. A graph is a simplicial complex where V is the vertex set of the graph and K consists of edges and vertices of the graph. A *filtration* is a collection of simplicial complexes $\{K_{\mathbf{x}}\}_{\mathbf{x}\in\mathbb{R}}$ indexed by the reals, with the property that $K_{\mathbf{x}} \subseteq K_{\mathbf{y}}$ for each $\mathbf{x} \leq \mathbf{y}$. Extending this definition to collections of complexes indexed by $\mathbb{R}^2$ with the product partial order ($\mathbf{x} \leq \mathbf{y} \in \mathbb{R}^2$ if $x_1 \leq y_1$ and $x_2 \leq y_2$), we get a *bifiltration*.

▶ **Definition 1** (Bifiltration). *A bifiltration is a collection of simplicial complexes* $\{K_{\mathbf{x}}\}_{\mathbf{x}\in\mathbb{R}^2}$ *where* $K_{\mathbf{x}} \subseteq K_{\mathbf{y}}$ *for all comparable* $\mathbf{x} \leq \mathbf{y} \in \mathbb{R}^2$.

Let K be a simplicial complex and $f\colon K \to \mathbb{R}^2$ be a map with the property that $f(\sigma) \leq f(\tau) \in \mathbb{R}^2$ for all $\sigma \subseteq \tau$. For each $\mathbf{x} \in \mathbb{R}^2$, let $K_{\mathbf{x}} \coloneqq \{\sigma \in K\colon f(\sigma) \leq \mathbf{x}\}$. Clearly, $K_{\mathbf{x}} \subseteq K_{\mathbf{y}}$ for all comparable $\mathbf{x} \leq \mathbf{y} \in \mathbb{R}^2$. The resulting bifiltration, denoted $\{K^f_{\mathbf{x}}\}_{\mathbf{x}\in\mathbb{R}^2}$, is called the *sub-level set* bifiltration of the *bifiltration function* f.

A filtration in the 1-parameter case induces a persistence module, obtained by considering the inclusion-induced linear maps between the vector spaces given by the homology groups of the simplicial complexes comprising the filtration. Similarly, we obtain a 2-parameter persistence module from a bifiltration.

▶ **Definition 2** (2-parameter persistence module). *A 2-parameter persistence module is an assignment of finite-dimensional vector spaces $M_{\mathbf{x}}$ for each $\mathbf{x} \in \mathbb{R}^2$, and of linear maps $M_{\mathbf{x}\leq\mathbf{y}} \colon M_{\mathbf{x}} \to M_{\mathbf{y}}$ for all comparable $\mathbf{x} \leq \mathbf{y} \in \mathbb{R}^2$, with the properties that $M_{\mathbf{x}\leq\mathbf{x}} = id$ and $M_{\mathbf{y}\leq\mathbf{z}} \circ M_{\mathbf{x}\leq\mathbf{y}} = M_{\mathbf{x}\leq\mathbf{z}}$ for all $\mathbf{x} \leq \mathbf{y} \leq \mathbf{z} \in \mathbb{R}^2$.*

Readers familiar with category theory can recognize that a 2-parameter persistence module is a functor $M \colon \mathbb{R}^2 \to \mathbf{vec}_{\mathbb{F}}$ where the poset $\mathbb{R}^2$ is regarded as a category and $\mathbf{vec}_{\mathbb{F}}$ denotes the category of finite-dimensional vector spaces over a field $\mathbb{F}$.

Given a bifiltration $\{K_{\mathbf{x}}\}_{\mathbf{x}\in\mathbb{R}^2}$, we consider $H_i(K_{\mathbf{x}})$, the ith homology group of the simplicial complex $K_{\mathbf{x}}$ over a field F, say $\mathbb{Z}_2$. Then, for each inclusion $K_{\mathbf{x}} \subseteq K_{\mathbf{y}}$, we get an induced linear map $H_i(K_{\mathbf{x}}) \to H_i(K_{\mathbf{y}})$. By this assignment, we get a 2-parameter persistence module.

Let $\{K^f_{\mathbf{x}}\}_{\mathbf{x}\in\mathbb{R}^2}$ be a sub-level set bifiltration. Then the 2-parameter persistence module M obtained by the homology assignment is called the 2-parameter persistence module induced by f, and we denote it as M_f.

As mentioned in Section 2, GRIL vectorization uses the concept of generalized rank introduced by [32], which we define below formally.

▶ **Definition 3** (Generalized Rank). *Let M be a 2-parameter persistence module. The restriction of M to an interval I of $\mathbb{R}^2$, denoted by $M|_I$, is the diagram formed by the collection of vector spaces $M_{\mathbf{x}}$ for $\mathbf{x} \in I$ and linear maps $M_{\mathbf{x}\leq\mathbf{y}}$ for all comparable $\mathbf{x} \leq \mathbf{y} \in I$. Then, the generalized rank of M over I is defined as the rank of the canonical map from the limit $\varprojlim M|_I$ to colimit $\varinjlim M|_I$:*

$$\mathsf{rk}^M(I) := rank\left(\varprojlim M|_I \to \varinjlim M|_I\right).$$

We refer the reader to [36] for the definitions of limit, colimit, and the construction of the canonical limit-to-colimit map. Intuitively, generalized rank captures the number of independent topological features supported over the interval I. It can be computed efficiently for 2-parameter persistence modules [18] using zigzag persistence [17].

▶ Remark 4. In the special case where I is a rectangle, $\mathsf{rk}^M(I)$ is the rank of the linear map $M_{\mathbf{u}\leq\mathbf{v}}$ from the lower left corner $\mathbf{u}$ of the rectangle to the upper right corner $\mathbf{v}$.

3.2 o-minimal geometry

We recall the definitions of o-minimal structure and definable function here.

▶ **Definition 5** (o-minimal structure). *An o-minimal structure on $\mathbb{R}$ is a collection $\{S_n\}_{n\in\mathbb{N}}$ where each S_n is a set of subsets of $\mathbb{R}^n$ such that:*

1. *S_1 is exactly the collection of finite union of points and intervals.*
2. *S_n contains all the sets of the form $\{\mathbf{x} \in \mathbb{R}^n \colon p(\mathbf{x}) = 0\}$ where p is a polynomial on $\mathbb{R}^n$.*
3. *S_n is a Boolean sub-algebra of $\mathbb{R}^n$ for all n.*
4. *If $A \in S_n$ and $B \in S_m$, then $A \times B \in S_{n+m}$.*
5. *If $\pi \colon \mathbb{R}^{n+1} \to \mathbb{R}^n$ is the canonical projection onto the first n-coordinates, then for $A \in S_{n+1}$, $\pi(A) \in S_n$.*

A subset $A \in S_n$ for some $n \in \mathbb{N}$ is known as a *definable set* in the o-minimal structure. Let $A \in S_n$ be given. A function $f \colon A \to \mathbb{R}^m$ is said to be *definable* if the graph of the function in $\mathbb{R}^{n+m}$ is a definable set.

We note that all semi-algebraic functions are definable. In fact, the author in, [47], shows that there exists an o-minimal structure that simultaneously contains all semi-algebraic sets and the graph of the exponential function. The authors in [14] use this result to show that a

neural network, being a composition of definable functions, is definable (Corollary 5.11 [14]). For readers not familiar with the notion of definable functions, it is sufficient to know that a piecewise affine map is definable.

3.3 Stochastic sub-gradient descent

In this subsection, we briefly review the result and the conditions under which stochastic sub-gradient descent converges as shown in [14]. In [7], authors use this result in the setting of 1-parameter persistent homology and show that stochastic sub-gradient descent converges in that case. We begin by reviewing the definition of Clarke subdifferential. Given a locally Lipschitz function $f\colon \mathbb{R}^n \to \mathbb{R}$, the Clarke subdifferential of f at a point x is defined as:

$$\partial f(x) \coloneqq \mathbf{Co}\left\{s \in \mathbb{R}^n \colon \exists x^k \to x, \nabla f(x^k) \text{ exists, and } \nabla f(x^k) \to s\right\},$$

where **Co** denotes the convex hull of the set.

Now, given a loss function $\mathcal{L}$ with the objective of minimizing it, consider the differential inclusion

$$\dot{\mathbf{z}}(t) \in -\partial\mathcal{L}(\mathbf{z}(t)) \quad \text{for almost every } t,$$

where $\partial\mathcal{L}(\mathbf{z}(t))$ denotes the Clarke subdifferential of the function $\mathcal{L}$ at the point $\mathbf{z}(t)$. The solutions $\mathbf{z}(t)$ are the trajectories of the sub-gradient of $\mathcal{L}$ which can be approximated by the standard stochastic sub-gradient descent method given by:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k(\mathbf{y}_k + \zeta_k), \quad \mathbf{y}_k \in \partial\mathcal{L}(\mathbf{x}_k), \tag{1}$$

where the sequence $(\alpha_k)_k$ is the learning rate and $(\zeta_k)_k$ is a sequence of random variables or "noise". [14] establish that under mild technical assumptions (stated below) on these two sequences, the stochastic sub-gradient method converges almost surely to critical points of $\mathcal{L}$ if $\mathcal{L}$ is locally Lipschitz and definable.

Consider the following assumptions (Assumption C in [14]):

1. for any k, $\alpha_k \geq 0$, $\Sigma_{k=1}^{\infty} = \infty$ and $\Sigma_{k=1}^{\infty}\alpha_k^2 < \infty$;
2. $\sup_k ||x_k|| < \infty$, almost surely;
3. denoting by $\mathcal{F}_k$ the increasing sequence of σ-algebras $\mathcal{F}_k = \sigma(x_j, y_j, \zeta_j, j < k)$, there exists a function $p\colon \mathbb{R}^d \to \mathbb{R}$ which is bounded on bounded sets such that almost surely, for any k,

$$\mathbb{E}\left[\zeta_k|\mathcal{F}_k\right] = 0 \text{ and } \mathbb{E}\left[||\zeta_k||^2|\mathcal{F}_k\right] < p(x_k).$$

▶ **Proposition 6** (Corollary 5.9 [14]). *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a locally Lipschitz definable function*[1]*. Consider the iterates $\{\mathbf{x}_k\}_{k\geq 1}$ produced by the stochastic sub-gradient method (Eq (1)) and suppose Assumption C of [14] holds. Then, almost surely, every limit point of the iterates $\{\mathbf{x}_k\}_{k\geq 1}$ is critical for f and the function values $\{f(\mathbf{x}_k)\}_{k\geq 1}$ converge.*

4 Differentiability of GRIL

We begin this section by proving that Gril is a piecewise affine map in section 4.1. Gril being piecewise affine leads to stochastic sub-gradient converging almost surely on Gril, discussed in section 4.2. In section 4.3, we give an explicit formula for differential of Gril.

[1] The authors in [14] show that the result holds for a more general class of C^d-stratifiable functions.

4.1 GRIL as a piecewise affine map

We begin by recalling the definitions of ℓ-worm and GRIL.

▶ **Definition 7** (discrete ℓ-worm, [48]). *Let* $\boxed{\mathbf{p}}_d \coloneqq \{\mathbf{w} : \|\mathbf{p} - \mathbf{w}\|_\infty \leq d\}$ *be the* d*-square centered at* $\mathbf{p} \in \mathbb{R}^2$ *with side* $2d$*. Given* $\ell \geq 1$ *and* $d > 0$*, the* ℓ-worm, $\boxed{\mathbf{p}}_d^\ell$*, is defined as the union of all* d*-squares* $\boxed{\mathbf{q}}_d$ *centered at some point* $\mathbf{q}$ *on the off-diagonal line segment* $\mathbf{p} \pm \alpha \cdot (1, -1)$ *with* $\alpha = j \cdot d$ *where* $j \in \{1, \ldots, \ell - 1\}$.

Refer to Figure 1 for a 2-worm.

▶ **Definition 8** (GRIL, [48]). *For a 2-parameter persistence module* M*, the Generalized Rank Invariant Landscape (GRIL) is a function* $\lambda^M : \mathbb{R}^2 \times \mathbb{N}_+ \times \mathbb{N}_+ \to \mathbb{R}$ *defined as*

$$\lambda^M(\mathbf{p}, k, \ell) \coloneqq \sup\left\{d \geq 0 \colon \mathsf{rk}^M\left(\boxed{\mathbf{p}}_d^\ell\right) \geq k\right\}.$$

Fixing the bifiltration function f and k, ℓ, the landscape λ^M provides a function $\lambda^{M_f}_{k,\ell} : \mathbb{R}^2 \to \mathbb{R}$, $\mathbf{p} \mapsto \lambda^{M_f}(\mathbf{p}, k, \ell)$. The GRIL vector is the vector of values of $\lambda^{M_f}_{k,\ell}$ evaluated at a set of chosen sample points $\{\mathbf{p}_1, \ldots, \mathbf{p}_s\} \subset \mathbb{R}^2$.

Let K be a simplicial complex with n simplices, labelled $\sigma_1, \sigma_2, \ldots, \sigma_n$. A bifiltration function $f \colon K \to \mathbb{R}^2$ can be viewed as a vector $\mathbf{v}_f \in \mathbb{R}^{2n}$ where, for $k = 1, \ldots, n$, $\mathbf{v}_f[2k-1] = f_x(\sigma_k)$ and $\mathbf{v}_f[2k] = f_y(\sigma_k)$. Here, $f_x(\sigma)$ and $f_y(\sigma)$ denote the x- and y-coordinates of the vector $f(\sigma)$ respectively. Notice that the vectors in $\mathbb{R}^{2n}$ that correspond to a valid bifiltration function form a convex cone. We work with this set of vectors in $\mathbb{R}^{2n}$. In this setting, the authors in [48] show that GRIL is Lipschitz continuous in the following sense.

▶ **Proposition 9.** *Let* X *be a discrete space with* $|X| = n$*. For fixed* $k, \ell, \mathbf{p}$*, let* $\Lambda^{\mathbf{p}}_{k,\ell} \colon \mathbb{R}^{2n} \to \mathbb{R}$ *be the map* $\mathbf{v}_f \mapsto \lambda^{M_f}_{k,\ell}(\mathbf{p})$*. Then,* $\Lambda^{\mathbf{p}}_{k,\ell}$ *is Lipschitz continuous.*

By Rademacher's theorem [23], $\Lambda^{\mathbf{p}}_{k,\ell}$ is differentiable almost everywhere.

Let $\mathcal{G}_{k,\ell} \colon \mathbb{R}^{2n} \to \mathbb{R}^s$ be the map defined as:

$$\mathcal{G}_{k,\ell}(v_f) = \left[\Lambda^{\mathbf{p}_1}_{k,\ell}(\mathbf{v}_f), \Lambda^{\mathbf{p}_2}_{k,\ell}(\mathbf{v}_f), \ldots, \Lambda^{\mathbf{p}_s}_{k,\ell}(\mathbf{v}_f)\right]^T \tag{2}$$

where $\{\mathbf{p}_j\}_{j=1}^s$ are the s sampled center points. We drop the k, ℓ and refer to $\mathcal{G}_{k,\ell}$ as $\mathcal{G}$ whenever k, ℓ are well understood. We note that $\mathcal{G}$ is also a function of the center points $\mathbf{p}_j$ for all j. We show that $\mathcal{G}$ is piecewise affine.

For notational convenience, in what follows we denote $f_x(\sigma)$ as σ^x and $f_y(\sigma)$ as σ^y, and we call them the *simplex coordinates* of σ. Similarly, we denote the x-coordinate and y-coordinate of $\mathbf{p}$ as $\mathbf{p}^x$ and $\mathbf{p}^y$ respectively.

We observe that, if there are two simplices σ_i, σ_j such that for some $\rho \in \mathbb{Z}$ and $0 \leq \rho \leq \ell$ and $a, b \in \{x, y\}$, one has $|\sigma_i^a - \mathbf{p}_t^a| = \rho \cdot |\sigma_j^b - \mathbf{p}_t^b|$ then, say for $a = x$ and $b = y$, the point representing the vector $\mathbf{v}_f$ lies on a hyperplane in $\mathbb{R}^{2n}$:

$$\left\{\mathbf{v} \in \mathbb{R}^{2n} \colon\ |\mathbf{v}[2i-1] - \mathbf{p}_t^x| = \rho \cdot |\mathbf{v}[2j] - \mathbf{p}_t^y|\right\}.$$

Corresponding to each such pair of simplex coordinates, we get one hyperplane. Here, in the example, we chose one simplex coordinate to be x-coordinate and the other to be y-coordinate. This, however, need not always be the case: we can have all four possible combinations of x- and y-coordinates, corresponding to each of which we get a hyperplane. We refer the reader to Appendix A in [40] for the exact formula of all such hyperplanes. Combining all

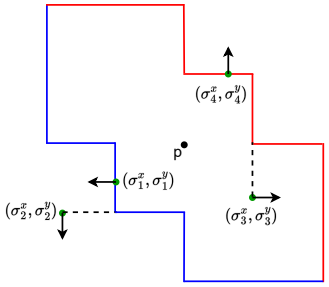


Figure 1 A 2-worm with lower boundary colored in blue and upper boundary colored in red. The figure also shows the possible cases of **constraining simplex coordinates**; σ_1 is a *lower x-constraining* simplex coordinate because the x-coordinate σ_1^x constrains the lower boundary of the worm and prevents the worm from expanding to the left; σ_2 shows a *lower y-constraining* simplex coordinate because σ_2^y also constrains the lower boundary of the worm and prevents the worm from expanding downwards. Similarly, σ_3 and σ_4 are *upper x-constraining* and *upper y-constraining* respectively. The arrows depict the gradient directions as described in Theorem 13.

these hyperplanes, we get an *arrangement $\mathcal{H}$ of hyperplanes* in $\mathbb{R}^{2n}$ [26]. The arrangement $\mathcal{H}$ partitions $\mathbb{R}^{2n}$ into *relatively open* r-cells, $r \in \{0, \ldots, 2n\}$. We observe that this arrangement induces an affine stratification $\mathcal{S}_{\mathcal{H}}$ (refer [34] for the formal definition) of $\mathbb{R}^{2n}$ where the r-dimensional strata are precisely the r-cells.

In the following we prove that $\mathcal{G}$ is a piecewise affine map relative to this arrangement, meaning that it is affine on each stratum of $\mathcal{S}_{\mathcal{H}}$. To this end, we introduce the notions of *upper boundary*, *lower boundary*, and *constraining simplex coordinate* for an ℓ-worm which will allow us to characterize the strata of $\mathcal{S}_{\mathcal{H}}$ in terms of conditions on the bifiltration function. We give an intuitive explanation of these concepts using Figure 1. We refer the reader to Appendix A in [40] for the formal definitions.

In Figure 1, a 2-worm is shown with its lower boundary colored in blue and its upper boundary in red. As depicted, the *lower boundary* consists of the part of the boundary that is in the lower half of the worm, while the *upper boundary* consists of the rest of the boundary, i.e., a point belongs to the lower boundary (resp. upper boundary) if the open lower-set (resp. upper-set) of the point does not intersect the worm.

For an intuitive understanding of *constraining simplex coordinate*, consider the Gril value d of an ℓ-worm centered at $\mathbf{p}$ for a given rank k. It follows from Definition 8 that there exists at least one simplex σ such that σ^x or σ^y prevents the worm from expanding any further to have a Gril value more than d at $\mathbf{p}$. The coordinate in question (either σ^x or σ^y) is called a *constraining simplex coordinate* for the ℓ-worm centered at $\mathbf{p}$. Here, by preventing the worm from expanding, we mean that if the worm were to expand then the value of the generalized rank over the worm would drop below the given value of rank k. Note that an ℓ-worm can have multiple constraining simplex coordinates, including some coming from the same simplex σ.

We can characterize the top-dimensional ($2n$-dimensional) strata of $\mathcal{S}_{\mathcal{H}}$ in terms of conditions on constraining simplex coordinates for ℓ-worms, and consequently, in terms of conditions on the bifiltration functions.

▶ **Proposition 10.** *The top-dimensional strata of $\mathcal{S}_{\mathcal{H}}$ consist precisely of those bifiltration functions that have a unique constraining simplex coordinate for each ℓ-worm.*

We state the main theorem of this subsection.

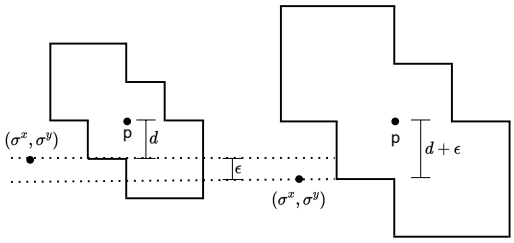


Figure 2 An intuitive understanding of the gradient assignment described in Theorem 13. The y-coordinate σ^y is at a distance of d from the y-coordinate of $\mathbf{p}$ in the left figure; σ is the only constraining simplex for the 2-worm. In the figure on the right, σ^y has *reduced* by ϵ and is now at a distance $d+\epsilon$ from $\mathbf{p}^y$. As a consequence, the value of GRIL increases from d to $d+\epsilon$. Thus, $\frac{\partial \Lambda^{\mathbf{P}}_{k,\ell}(\mathbf{v}_f)}{\partial \sigma^y} = -1$.

▶ **Theorem 11.** *Let K be a simplicial complex with n simplices. Let $k, \ell \in \mathbb{N}$, and let $\{\mathbf{p}_j\}_{j=1}^s$ be the s sampled center points for the ℓ-worms. Then, $\mathcal{G}$, as defined in Eq.*(2)*, is a piecewise affine map relative to the arrangement $\mathcal{H}$.*

Overview of the proof: $\mathcal{G}$ depends affinely on the simplex coordinates in each top-dimensional stratum, because there is a unique constraining simplex coordinate for each ℓ-worm. By continuity of $\mathcal{G}$ (Proposition 9), the restriction of $\mathcal{G}$ to each (affine) lower dimensional stratum is affine.

4.2 Stochastic sub-gradient descent

In our machine learning pipeline, the GRIL map $\mathcal{G}$ is post-composed with a loss function $N\colon \mathbb{R}^s \to \mathbb{R}$, derived e.g. from some neural network. In the corollary below, we give sufficient conditions on N that ensure the convergence of stochastic sub-gradient descent on $N \circ \mathcal{G}$.

▶ **Corollary 12.** *If $N\colon \mathbb{R}^s \to \mathbb{R}$ is definable and locally Lipschitz continuous, then, under the assumptions of Proposition 6 and Theorem 11, stochastic sub-gradient descent on $N \circ \mathcal{G}$ converges almost surely to critical points of $N \circ \mathcal{G}$.*

Overview of the proof: In the discussion towards the end of section 3.3, we saw that any piecewise affine map is definable. Since $\mathcal{G}$ is piecewise affine, it is locally Lipschitz and definable. We know that the composition of two definable functions is definable, and that the composition of two locally Lipschitz functions is locally Lipschitz. These facts, put together with Proposition 6, immediately lead to the fact that stochastic sub-gradient descent converges almost surely to critical points of $N \circ \mathcal{G}$.

4.3 Differential of GRIL

In order to back-propagate gradients through the GRIL computation, we need an explicit formula for the differential of $\mathcal{G}$. Such a formula is needed only in the top-dimensional strata, where $\mathcal{G}$ is actually smooth. In lower-dimensional strata, sub-gradients can be approximated using, e.g., gradient sampling [3].

In section 4.1, we saw that $\mathcal{G}$ is piecewise affine, which implies that its differential is constant in each top-dimensional stratum. In order to compute it, we need to determine the sign of the gradient at different simplex coordinates. To do so, given a worm, we distinguish between simplex coordinates that constrain the lower boundary of the worm and those that constrain its upper boundary. Since each simplex has two coordinates, x and y, and these can

be constraining either the upper boundary or the lower boundary, we get four different cases: *lower x-constraining, lower y-constraining, upper x-constraining* and *upper y-constraining.* All these cases are shown in Figure 1.

We deduce an explicit formula for the differential of $\mathcal{G}$ in the top-dimensional strata.

▶ **Theorem 13.** *Let K be a simplicial complex with n simplices. Let $k, \ell \in \mathbb{N}$ and $\{\mathbf{p}_j\}_{j=1}^{s}$ be the s sampled center points for the ℓ-worms. Then, the differential of $\mathcal{G}$ at any $\mathbf{v}_f$ in a top-dimensional stratum is given by:*

$$\begin{pmatrix} \frac{\partial \Lambda^{\mathbf{p}_1}_{k,\ell}(\mathbf{v}_f)}{\partial \sigma^x_1} & \frac{\partial \Lambda^{\mathbf{p}_1}_{k,\ell}(\mathbf{v}_f)}{\partial \sigma^y_1} & \frac{\partial \Lambda^{\mathbf{p}_1}_{k,\ell}(\mathbf{v}_f)}{\partial \sigma^x_2} & \cdots & \frac{\partial \Lambda^{\mathbf{p}_1}_{k,\ell}(\mathbf{v}_f)}{\partial \sigma^y_n} \\ \vdots & & & & \vdots \\ \frac{\partial \Lambda^{\mathbf{p}_s}_{k,\ell}(\mathbf{v}_f)}{\partial \sigma^x_1} & \cdots & \cdots & & \frac{\partial \Lambda^{\mathbf{p}_s}_{k,\ell}(\mathbf{v}_f)}{\partial \sigma^y_n} \end{pmatrix}_{s \times 2n}$$

where,

$$\frac{\partial \Lambda^{\mathbf{p}_j}_{k,\ell}(\mathbf{v}_f)}{\partial \sigma^x_i} = \begin{cases} -1, & \text{if } \sigma_i \text{ is lower } x\text{-constraining for } \boxed{\mathbf{p}_j}^{\ell}_{d_j}, \\ +1, & \text{if } \sigma_i \text{ is upper } x\text{-constraining for } \boxed{\mathbf{p}_j}^{\ell}_{d_j}, \\ 0, & \text{otherwise,} \end{cases}$$

$$\frac{\partial \Lambda^{\mathbf{p}_j}_{k,\ell}(\mathbf{v}_f)}{\partial \sigma^y_i} = \begin{cases} -1, & \text{if } \sigma_i \text{ is lower } y\text{-constraining for } \boxed{\mathbf{p}_j}^{\ell}_{d_j} \\ +1, & \text{if } \sigma_i \text{ is upper } y\text{-constraining for } \boxed{\mathbf{p}_j}^{\ell}_{d_j} \\ 0, & \text{otherwise,} \end{cases}$$

and d_j is the GRIL *value at $\mathbf{p}_j$.*

Recall that each $\mathbf{v}_f$ in a top-dimensional stratum of $\mathcal{S}_{\mathcal{H}}$ corresponds to a bifiltration function f with a unique constraining simplex. Refer to Figure 2 for an intuitive explanation about the sign of the gradient and the different cases in Theorem 13, as shown by the arrows in Figure 1.

▶ **Corollary 14.** *Given the conditions of Theorem 13, partial derivatives of $\mathcal{G}$ with respect to $\mathbf{p}^x_j$ and $\mathbf{p}^y_j$ also exist and are given by:*

$$\frac{\partial \Lambda^{\mathbf{p}_j}_{k,\ell}(\mathbf{v}_f)}{\partial \mathbf{p}^x_j} = \begin{cases} +1, & \text{if there exists a lower } x\text{-constraining} \\ & \text{simplex for } \boxed{\mathbf{p}_j}^{\ell}_{d_j}, \\ -1, & \text{if there exists an upper } x\text{-constraining} \\ & \text{simplex for } \boxed{\mathbf{p}_j}^{\ell}_{d_j}, \\ 0, & \text{otherwise,} \end{cases}$$

$$\frac{\partial \Lambda^{\mathbf{p}_j}_{k,\ell}(\mathbf{v}_f)}{\partial \mathbf{p}^y_j} = \begin{cases} +1, & \text{if there exists a lower } y\text{-constraining} \\ & \text{simplex for } \boxed{\mathbf{p}_j}^{\ell}_{d_j}, \\ -1, & \text{if there exists an upper } y\text{-constraining} \\ & \text{simplex for } \boxed{\mathbf{p}_j}^{\ell}_{d_j}, \\ 0, & \text{otherwise.} \end{cases}$$

5 Experimental setup

In this section, we outline the overall experimental setup. We first describe some practical considerations involved in implementing D-Gril, and then describe the architecture used across all experiments.

5.1 Practical considerations

In section 4.1 we saw that $\mathcal{G}$ is piecewise affine and in section 4.2 we deduced that stochastic sub-gradient descent converges almost surely to critical points of $N \circ \mathcal{G}$ for any definable and locally Lipschitz loss function N. We gave an explicit formula for the differential of Gril in the top-dimensional strata in section 4.3, and at the beginning of that section we argued that gradient sampling can be used to approximate sub-gradients in lower-dimensional strata. In practice, for the sake of computational efficiency, we use a simple variant of gradient sampling, which consists of sampling a single nearby point and taking its gradient.

So far, we considered a bifiltration function on the entire simplicial complex K. However, in practice, we may need to extend the filtration function to the entire simplicial complex based on filtration function values on certain simplices. For example, in a lower-star filtration, the filtration function values on the higher dimensional simplices are dictated by the filtration function values on the vertices of the simplicial complex. In a Rips filtration, the filtration function values on the vertices are 0, while the filtration function values on higher dimensional simplices are dictated by the filtration function values on the edges.

In such scenarios, the bifiltration function f is given on a subcomplex L of the simplicial complex K. This is extended to a bifiltration function $\bar{f}$ on the entire simplicial complex in a piecewise constant manner. We use the example of lower-star bifiltration to show how our framework fits to this type of scenario. Let $f\colon K^0 \to \mathbb{R}^2$ be a function on the vertices of the simplicial complex K with m vertices and n simplices. The map f is extended to a piecewise constant map $\bar{f}\colon K \to \mathbb{R}^2$ as follows: for an edge $e = (u, v)$, we let $\bar{f}_x(e) = \max\{f_x(u), f_x(v)\}$ and $\bar{f}_y(e) = \max\{f_y(u), f_y(v)\}$; the values on higher dimensional simplices are defined inductively. Notice that for each simplex $\sigma \in K$, there is a *maximal x-vertex* and a *maximal y-vertex*, the vertices that give the x and y-values to the simplex respectively. We know that f can be represented as a vector $\mathbf{v}_f \in \mathbb{R}^{2m}$ and $\bar{f}$ as a vector $\mathbf{v}_{\bar{f}} \in \mathbb{R}^{2n}$ after ordering the simplices of K. Consider the map $q\colon \mathbb{R}^{2m} \to \mathbb{R}^{2n}$ given by $q(\mathbf{v}_f) = \mathbf{v}_{\bar{f}}$. The map q is piecewise affine and thus, $\mathcal{G} \circ q$ is piecewise affine as well. Therefore, we can apply stochastic sub-gradient descent on $N \circ \mathcal{G} \circ q$ with analogous convergence guarantees.

Observe that we get an arrangement of hyperplanes on $\mathbb{R}^{2m}$, $\mathcal{H}_q$, analogous to $\mathcal{H}$ on $\mathbb{R}^{2n}$ that we saw in the previous subsections. On the top-dimensional cells of $\mathcal{H}_q$, the differential of q is well-defined and constant and expressions analogous to the ones in Theorem 13 can be derived, and the chain rule can be used for back-propagating gradients. As mentioned previously, we use a simple form of gradient sampling to approximate sub-gradients when the gradient is not well-defined. For the experimental section that follows, N is a neural network with one of the standard choices for loss functions, which is definable as discussed towards the end of section 3.2.

We observe that if the bifiltration f is extended to $\bar{f}$ as described above , then we end up in a scenario where multiple simplices have the same x or y-values. This is a degenerate scenario with respect to the assumptions in Theorem 11. To avoid this, in practice, we add an infinitesimal perturbation to the higher dimensional simplices to ensure the values are not exactly the same. We note that this is, in spirit, similar to gradient sampling.

5.2 Model architecture

Every instance in each dataset is an attributed graph with a label associated with it. We use a Graph Isomorphism Network (GIN) [49], as a candidate GNN architecture, to obtain a bifiltration function over the vertices. We consider the lower-star bifiltration of this function. This gives us a bifiltration function on the entire graph, which we denote as f. Let $\mathbf{v}_f$ denote the vector representation of f which we get after ordering the n simplices of the graph. We divide the range of f into a grid of size 100×100. We randomly initialize the s center points from this grid and compute the vector $\mathcal{G}_{k,\ell}(\mathbf{v}_f)$ (Eq. (2)) for $k = 1, 2, 3$ and $\ell = 2$ at these center points. This vector $\mathcal{G}_{k,\ell}(\mathbf{v}_f)$ is fed to a 3-layer Multilayer Perceptron (MLP) classifier and the loss is back-propagated through the GRIL computation according to the gradient described in Section 4. Refer to Figure 3 for an illustration. Note that the positions of the s center points are also optimized as the gradient can be back-propagated to the coordinates of the center points as well. We refer the reader to Appendix B in [40] for additional details.

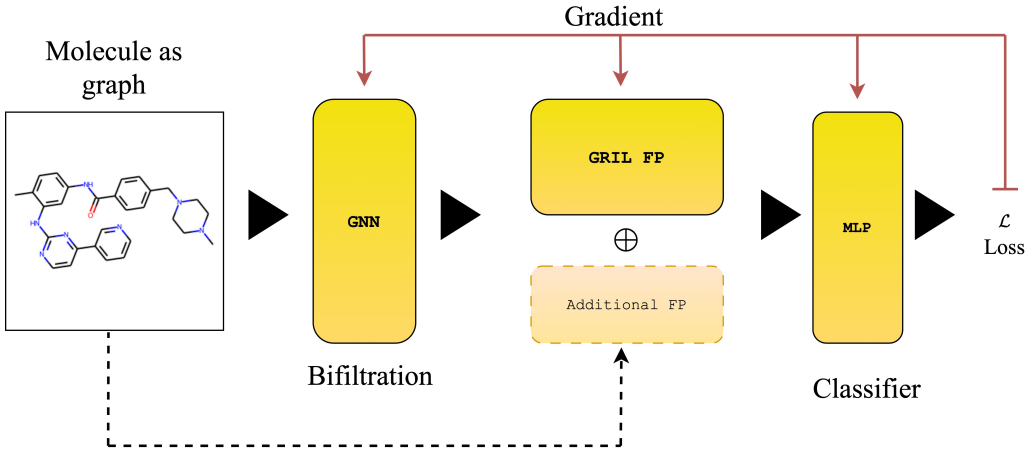


Figure 3 Architecture choice for bio-activity prediction; the bifiltration fuction f is learnt compared to the standard multiparameter pipeline; $\oplus$ denotes concatenation of vectors.

6 Experimental results

In this section, we present the experimental results on various bio-activity prediction datasets and benchmark graph datasets. We begin the section with a brief description of bio-activity prediction datasets followed by experimental results on those datasets. Then, in section 6.2, we show that D-GRIL can be, more generally, applied in the context of bifiltration learning on standard benchmark graph datasets. We compare with existing multiparameter persistent homology methods on these datasets. All the reported accuracy/ROC-AUC scores, in this section, are 5-fold cross-validated scores.

6.1 Bio-activity prediction datasets

The data is extracted from the ChEMBL database [24, 13]. Each dataset contains the SMILES [46] encoding of a novel drug (compound), and activity pairs for a target of interest. For example, the EGFR dataset contains all the molecules that have been tested against epidermal growth factor receptor (EGFR) kinase, and their measured bio-activity. The

Table 1 Test ROC-AUC on ChEMBL datasets. D-GRIL performs better than GIN with *sum* pooling and GRIL with bifiltration obtained from pre-trained GIN.

Dataset	GIN	GIN-GRIL	GIN-MPSM	D-GRIL
EGFR	55.60 ± 8.61	58.39 ± 2.51	$\mathbf{66.97 \pm 1.87}$	65.38 ± 5.34
ERBB2	57.06 ± 8.58	61.28 ± 3.66	68.75 ± 1.98	$\mathbf{69.24 \pm 5.06}$
CHEMBL1163125	58.48 ± 4.37	54.13 ± 1.09	61.23 ± 2.47	$\mathbf{65.69 \pm 1.57}$
CHEMBL2148	52.88 ± 0.91	50.24 ± 0.18	51.45 ± 0.87	$\mathbf{53.13 \pm 3.70}$
CHEMBL4005	55.90 ± 4.63	51.85 ± 3.55	$\mathbf{61.35 \pm 1.49}$	59.34 ± 5.70

SMILES encodings of the molecules are then converted to graphs with MOLFEAT [41]. Bio-activity is measured in *half maximal inhibitory concentration* (IC_{50}), which qualitatively indicates how much a drug is needed, *in vitro*, to inhibit a particular process by 50%. To facilitate the comparison of IC_{50} values, it is common practice to convert IC_{50} to $pIC_{50} = -\log_{10}(IC_{50})$, expressed in molar units. Threshold for activity cutoff $pIC_{50} = 6.3$ is used throughout the paper. We refer the reader to Appendix B in [40] for additional details.

In the first set of experiments, we compare D-GRIL with (i) a standard GNN model– GIN with sum-pooling, (ii) with GIN and GRIL as a readout layer and with (iii) GIN and MPSM [44]. Note that for the results on GIN-GRIL, GRIL is used as a passive readout layer, i.e., we obtain a bifiltration function from a pre-trained GIN (pre-trained for graph classification on the same dataset) and compute GRIL on it. We use the exact same pipeline as GIN-GRIL and train it end-to-end using D-GRIL and GIN-MPSM and report the results. The classifier, 3-layer MLP, is the same for all the experiments. Refer to Figure 3 for the pipeline. From Table 1, we can see that adding topological information after GIN has been trained can improve the performance in some cases and need not be beneficial in others. However, it is clear that adding topological information in an end-to-end framework appears to be beneficial for bio-activity prediction. D-GRIL has comparable performance to MPSM.

Further, we perform a series of experiments where we augment other popular molecular fingerprints such as ECFP and Morgan3 [43, 37] with the D-GRIL framework. In these experiments, we compare the performance of the model with and without D-GRIL augmentation. We report the results in Table 2. We see from the table that augmenting with D-GRIL seems to, generally, increase the performance of the model. Fingerprints such as ECFP and Morgan3 can essentially represent an infinite number of different molecular features, including stereochemical information [43] but they are not effective in capturing global features of molecules such as size and shape [4]. Combining D-GRIL with these fingerprints augments the model with topological information, yielding an improved performance.

6.2 Benchmark graph datasets

D-GRIL can be used more generally for filtration learning on graph datasets. We perform a series of experiments with benchmark graph datasets such as MUTAG, PROTEINS, DHFR, COX2 [38] and compare with existing multiparameter persistence methods. Details about the datasets are given in Appendix B in [40]. We report the results in Table 3. For a valid comparison, we use a 3-layer MLP as the classifier for all the multiparameter signatures. Hence, the numbers reported look different from the ones in [48], [6] where the authors consider XGBoost [9]. In fact, in [48], the authors show that XGBoost performs much better than a 3-layer MLP classifier for GRIL vectors on these datasets. We can see from the table that learning the bifiltration function seems to perform better than multiparameter

Table 2 Test ROC-AUC scores on ChEMBL datasets; augmenting ECFP and Morgan3 fingerprints with D-GRIL increases the classification performance for most of the datasets.

Dataset	ECFP	ECFP+D-GRIL	Morgan3	Morgan3+D-GRIL
EGFR	83.27 ± 1.10	$\mathbf{84.37 \pm 1.22}$	82.39 ± 1.35	$\mathbf{83.57 \pm 1.40}$
ERBB2	83.53 ± 1.31	$\mathbf{86.24 \pm 2.27}$	83.19 ± 1.26	$\mathbf{85.51 \pm 1.47}$
CHEMBL1163125	83.94 ± 1.23	$\mathbf{86.33 \pm 0.67}$	83.55 ± 1.20	$\mathbf{84.85 \pm 0.82}$
CHEMBL203	81.74 ± 0.90	$\mathbf{81.86 \pm 0.90}$	80.85 ± 1.32	$\mathbf{81.64 \pm 0.45}$
CHEMBL2148	73.96 ± 2.59	$\mathbf{74.13 \pm 2.33}$	72.79 ± 1.97	$\mathbf{74.50 \pm 2.61}$
CHEMBL279	76.72 ± 1.34	$\mathbf{78.75 \pm 0.77}$	76.76 ± 0.50	$\mathbf{78.05 \pm 1.18}$
CHEMBL2815	73.69 ± 1.53	$\mathbf{76.51 \pm 1.48}$	73.42 ± 0.56	$\mathbf{75.22 \pm 2.85}$
CHEMBL4005	$\mathbf{80.45 \pm 1.30}$	80.41 ± 0.97	79.95 ± 1.30	$\mathbf{80.55 \pm 0.82}$
CHEMBL4722	78.05 ± 1.64	$\mathbf{78.51 \pm 1.83}$	78.76 ± 1.37	$\mathbf{79.17 \pm 1.30}$

Table 3 Test accuracies of D-GRIL on benchmark graph datasets.

Dataset	D-GRIL	MPSM	GRIL	MP-I	MP-L	MP-K	P
MUTAG	$\mathbf{85.09 \pm 5.99}$	78.44 ± 3.33	83.49 ± 3.64	74.99 ± 2.79	82.42 ± 3.72	79.27 ± 2.45	66.50 ± 0.87
PROTEINS	69.45 ± 4.11	68.37 ± 3.31	66.31 ± 2.34	70.80 ± 3.09	$\mathbf{70.80 \pm 1.31}$	61.70 ± 2.98	59.57 ± 0.08
DHFR	61.24 ± 4.37	60.97 ± 1.98	$\mathbf{61.64 \pm 1.66}$	60.98 ± 0.10	61.11 ± 0.25	60.98 ± 0.10	60.98 ± 0.10
COX2	78.16 ± 0.41	77.73 ± 0.36	78.16 ± 0.41	78.16 ± 0.41	78.16 ± 0.41	78.16 ± 0.41	78.16 ± 0.41
IMDB-BINARY	$\mathbf{62.60 \pm 6.56}$	60.80 ± 3.82	50.00 ± 0.00	56.60 ± 2.94	50.00 ± 0.00	50.30 ± 1.12	50.00 ± 0.00

Table 4 Reported times (hh:mm:ss) are training times per fold averaged over 5 folds that we used for training. All the experiments have been performed on a machine with AMD EPYC 7313 16-Core Processor and NVIDIA A10 GPU.

Dataset	D-GRIL	MPSM
MUTAG	00:05:08	00:08:47
COX2	00:11:34	00:18:32
DHFR	00:16:48	00:27:12
PROTEINS	00:30:03	00:50:26

persistence methods on popular choices of bifiltration functions on most datasets. We can also see that D-GRIL performs better than GRIL, supporting our argument for an end-to-end learning framework. We also see that D-GRIL performs better than MPSM. The training times for the two end-to-end learning methods, D-GRIL and MPSM are reported in Table 4. We can see that the training times for D-GRIL are improved by about 40% compared to the ones of MPSM. We refer the reader to Figure 4 [40] for visualizations of bifiltration functions learnt by D-GRIL.

Additionally, we performed experiments where we added a fusion layer to combine the representation power of both D-GRIL and MPSM. We present the results in Table 5. We see from the table that combining the representations leads to a better overall performance, which includes a generally higher mean and a lower standard deviation. Thus, D-GRIL can also be used with existing vectorization methods in a complementary manner to capture additional topological information.

Table 5 Test accuracies of D-Gril+Mpsm on benchmark graph datasets.

Dataset	D-GRIL	MPSM	D-GRIL + MPSM
MUTAG	85.09 ± 5.99	78.44 ± 3.33	83.68 ± 3.06
PROTEINS	69.45 ± 4.11	68.37 ± 3.31	70.23 ± 3.56
DHFR	61.24 ± 4.37	60.97 ± 1.98	62.83 ± 3.68
COX2	78.16 ± 0.41	77.73 ± 0.36	78.72 ± 0.38
IMDB-BINARY	62.60 ± 6.56	60.80 ± 3.82	64.70 ± 3.96

7 Conclusions

In this paper, we propose a differentiable framework using Gril. We show that stochastic sub-gradient descent converges almost surely on Gril, and we compute an explicit formula for the differential of Gril. We use this formula to back-propagate gradients through the Gril computation, yielding the differentiable framework D-Gril. We use D-Gril to show that adding topological information in an end-to-end manner is beneficial for bio-activity prediction. Further, we show that D-Gril can be used in a more general setting, for filtration learning on graphs. Moreover, we show that D-Gril is faster and lighter, both in theory and in practice, than Mpsm. Our results indicate that learnt bifiltration functions on graphs, generally, give a better performance than the standard choices for bifiltration functions. Recently, the Gril framework has been extended to a variant of 2-parameter persistence called *quasi-zigzag* persistence [19]. The results in this paper may shed light on the question of differentiability in the quasi-zigzag framework.

References

1 Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017. URL: http://jmlr.org/papers/v18/16-337.html.

2 Peter Bubenik. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.*, 16:77–102, 2015. doi:10.5555/2789272.2789275.

3 James V Burke, Frank E Curtis, Adrian S Lewis, Michael L Overton, and Lucas EA Simões. Gradient sampling methods for nonsmooth optimization. *Numerical nonsmooth optimization: State of the art algorithms*, pages 201–225, 2020. URL: https://link.springer.com/chapter/10.1007/978-3-030-34910-3_6.

4 Alice Capecchi, Daniel Probst, and Jean-Louis Reymond. One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome. *Journal of Cheminformatics*, 12(1):43, June 2020. doi:10.1186/s13321-020-00445-4.

5 Gunnar Carlsson, Gurjeet Singh, and Afra Zomorodian. Computing multidimensional persistence. In *International Symposium on Algorithms and Computation*, pages 730–739. Springer, 2009. doi:10.1007/978-3-642-10631-6_74.

6 Mathieu Carrière and Andrew Blumberg. Multiparameter persistence image for topological machine learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22432–22444. Curran Associates, Inc., 2020. URL: https://proceedings.neurips.cc/paper/2020/file/fdff71fcab656abfbefaabecab1a7f6d-Paper.pdf.

7 Mathieu Carrière, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hariprasad Kannan, and Yuhei Umeda. Optimizing persistent homology based functions. In *International conference on machine learning*, pages 1294–1303. PMLR, 2021. URL: http://proceedings.mlr.press/v139/carriere21a/carriere21a.pdf.

8 Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Theo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2786–2796. PMLR, 26–28 August 2020. URL: https://proceedings.mlr.press/v108/carriere20a.html.

9 Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2939672.2939785.

10 Yuzhou Chen, Ignacio Segovia-Dominguez, Cuneyt Gurcan Akcora, Zhiwei Zhen, Murat Kantarcioglu, Yulia Gel, and Baris Coskunuzer. Emp: Effective multidimensional persistence for graph representation learning. In *The Second Learning on Graphs Conference*, 2023. URL: https://openreview.net/pdf?id=WScCJnX4ek.

11 René Corbet, Ulderico Fugacci, Michael Kerber, Claudia Landi, and Bei Wang. A kernel for multi-parameter persistent homology. *Computers & Graphics: X*, 2:100005, 2019. doi:10.1016/j.cagx.2019.100005.

12 Baris Coskunuzer and Cüneyt Gürcan Akçora. Topological methods in machine learning: A tutorial for practitioners, 2024. doi:10.48550/arXiv.2409.02901.

13 Mark Davies, Michał Nowotka, George Papadatos, Nathan Dedman, Anna Gaulton, Francis Atkinson, Louisa Bellis, and John P Overington. ChEMBL web services: streamlining access to drug discovery data and utilities. *Nucleic acids research*, 43(W1):W612–W620, 2015. doi:10.1093/NAR/GKV352.

14 Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D Lee. Stochastic subgradient method converges on tame functions. *Foundations of computational mathematics*, 20(1):119–154, 2020. doi:10.1007/S10208-018-09409-5.

15 Nima Dehmamy, Albert-Laszlo Barabasi, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/73bf6c41e241e28b89d0fb9e0c82f9ce-Paper.pdf.

16 Andac Demir, Baris Coskunuzer, Yulia Gel, Ignacio Segovia-Dominguez, Yuzhou Chen, and Bulent Kiziltan. Todd: Topological compound fingerprinting in computer-aided drug discovery. *Advances in Neural Information Processing Systems*, 35:27978–27993, 2022. URL: https://openreview.net/pdf?id=8hs7qlWcnGs.

17 Tamal K. Dey and Tao Hou. Fast computation of zigzag persistence. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 43:1–43:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.43.

18 Tamal K. Dey, Woojin Kim, and Facundo Mémoli. Computing generalized rank invariant for 2-parameter persistence modules via zigzag persistence and its applications. *Discret. Comput. Geom.*, 71(1):67–94, 2024. doi:10.1007/S00454-023-00584-Z.

19 Tamal K. Dey and Shreyas N. Samaga. Quasi zigzag persistence: A topological framework for analyzing time-varying data. In *Proceedings of the 1st Conference on Topology, Algebra, and Geometry in Data Science(TAG-DS 2025)*, volume 321 of *Proceedings of Machine Learning Research*, pages 147–165. PMLR, 01–02 December 2026. URL: https://proceedings.mlr.press/v321/dey26a.html.

20 Tamal K. Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022. doi:10.1017/9781009099950.

21 Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002. doi:10.1007/S00454-002-2885-2.

22 Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. Applied Mathematics. American Mathematical Society, 2010.

23 Herbert Federer. *Geometric measure theory*. Springer, 2014.

24 Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012. URL: https://academic.oup.com/nar/article/40/D1/D1100/2903401?login=false.

25 Barbara Giunti, Jānis Lazovskis, and Bastian Rieck. Donut: Database of original & non-theoretical uses of topology, 2022. URL: https://donut.topology.rocks.

26 J. E. Goodman and J. O'Rourke. *Handbook of Discrete and Computational Geometry, Second Edition*. Graduate Texts in Mathematics, Vol. 5. CRC Press, Boca Raton, FL, 2004.

27 Christoph D. Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4314–4323. PMLR, 2020. URL: http://proceedings.mlr.press/v119/hofer20b.html.

28 Christoph D Hofer, Roland Kwitt, and Marc Niethammer. Learning representations of persistence barcodes. *J. Mach. Learn. Res.*, 20(126):1–45, 2019. URL: https://www.jmlr.org/papers/volume20/18-358/18-358.pdf.

29 Christoph D. Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1634–1644, 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/883e881bb4d22a7add958f2d6b052c9f-Paper.pdf.

30 Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten M. Borgwardt. Topological graph neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=oxxUMeFwEHd.

31 Kwangho Kim, Jisu Kim, Manzil Zaheer, Joon Kim, Frédéric Chazal, and Larry Wasserman. PLLay: Efficient topological layer based on persistent landscapes. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15965–15977. Curran Associates, Inc., 2020. URL: https://proceedings.neurips.cc/paper/2020/file/b803a9254688e259cde2ec0361c8abe4-Paper.pdf.

32 Woojin Kim and Facundo Mémoli. Generalized persistence diagrams for persistence modules over posets. *Journal of Applied and Computational Topology*, 5(4):533–581, December 2021. doi:10.1007/s41468-021-00075-1.

33 Michael Lesnick and Matthew Wright. Interactive visualization of 2-d persistence modules. *CoRR*, abs/1512.00180, 2015. arXiv:1512.00180.

34 Jacob Leygonie, Steve Oudot, and Ulrike Tillmann. A framework for differential calculus on persistence barcodes. *Foundations of Computational Mathematics*, pages 1–63, 2021. URL: https://link.springer.com/article/10.1007/s10208-021-09522-y.

35 David Loiseaux, Luis Scoccola, Mathieu Carrière, Magnus Bakke Botnan, and Steve Oudot. Stable vectorization of multiparameter persistent homology using signed barcodes as measures. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/d75c474bc01735929a1fab5d0de3b189-Abstract-Conference.html.

36 Saunders MacLane. *Categories for the working mathematician*. Graduate Texts in Mathematics, Vol. 5. Springer-Verlag, New York-Berlin, 1971.

37 Harry L Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of chemical documentation*, 5(2):107–113, 1965. URL: https://pubs.acs.org/doi/pdf/10.1021/c160017a018.

38 Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. arXiv:2007.08663.

39 Soham Mukherjee, Shreyas N. Samaga, Cheng Xin, Steve Oudot, and Tamal K. Dey. D-GRIL: End-to-End Topological Learning with 2-parameter Persistence. Software, swhId: swh:1:dir:379b8266de5d11b54b8041b9a4af9b9dde1fa254 (visited on 2026-04-13). URL: https://github.com/TDA-Jyamiti/d-gril/.

40 Soham Mukherjee, Shreyas N. Samaga, Cheng Xin, Steve Oudot, and Tamal K. Dey. D-gril: End-to-end topological learning with 2-parameter persistence, 2025. doi:10.48550/arXiv.2406.07100.

41 Emmanuel Noutahi, Cas Wognum, Hadrien Mary, Honoré Hounwanou, Kyle M. Kovary, Desmond Gilmour, thibaultvarin r, Jackson Burns, Julien St-Laurent, t, DomInvivo, Saurav Maheshkar, and rbyrne momatx. datamol-io/molfeat: 0.9.4, September 2023. doi:10.5281/zenodo.8373019.

42 Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4741–4748, 2015. doi:10.1109/CVPR.2015.7299106.

43 David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010. doi:10.1021/CI100050T.

44 Luis Scoccola, Siddharth Setlur, David Loiseaux, Mathieu Carrière, and Steve Oudot. Differentiability and optimization of multiparameter persistent homology. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 43986–44011. PMLR, 21–27 July 2024. URL: https://proceedings.mlr.press/v235/scoccola24a.html.

45 Oliver Vipond. Multiparameter persistence landscapes. *Journal of Machine Learning Research*, 21(61):1–38, 2020. URL: http://jmlr.org/papers/v21/19-054.html.

46 David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988. doi:10.1021/CI00057A005.

47 Alex Wilkie. Model completeness results for expansions of the ordered field of real numbers by restricted pfaffian functions and the exponential function. *Journal of the American Mathematical Society*, 9(4):1051–1094, 1996. URL: https://community.ams.org/journals/jams/1996-09-04/S0894-0347-96-00216-0/S0894-0347-96-00216-0.pdf.

48 Cheng Xin, Soham Mukherjee, Shreyas N. Samaga, and Tamal K. Dey. GRIL: a 2-parameter persistence based vectorization for machine learning. In *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*, volume 221 of *Proceedings of Machine Learning Research*, pages 313–333. PMLR, July 2023. URL: https://proceedings.mlr.press/v221/xin23a.html.

49 Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km.

50 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 347–356, 2004. doi:10.1145/997817.997870.

Hardness of High-Dimensional Linear Classification

Alexander Munteanu
TU Dortmund, Germany

Simon Omlor
TU Dortmund, Germany

Jeff M. Phillips
University of Utah, Salt Lake City, UT, USA

Abstract

We establish new exponential in dimension lower bounds for the Maximum Halfspace Discrepancy problem, which models linear classification. Both are fundamental problems in computational geometry and machine learning in their exact and approximate forms. However, only $O(n^d)$ and respectively $\tilde{O}(1/\varepsilon^d)$ upper bounds are known and complemented by polynomial lower bounds that do not support the exponential in dimension dependence. We close this gap up to polylogarithmic terms by reduction from widely-believed hardness conjectures for Affine Degeneracy testing and k-Sum problems. Our reductions yield matching lower bounds of $\tilde{\Omega}(n^d)$ and respectively $\tilde{\Omega}(1/\varepsilon^d)$ based on Affine Degeneracy testing, and $\tilde{\Omega}(n^{d/2})$ and respectively $\tilde{\Omega}(1/\varepsilon^{d/2})$ conditioned on k-Sum. The first bound also holds unconditionally if the computational model is restricted to make *sidedness* queries, which corresponds to a widely spread setting implemented and optimized in many contemporary algorithms and computing paradigms.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Conditional Hardness, k-Sum, Affine Degeneracy, Halfspace Discrepancy, Classification

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.80

Related Version *arXiv Preprint*: `https://arxiv.org/abs/2603.19061` [50]

Funding *Alexander Munteanu*: supported by the German Research Foundation (DFG) - grant MU 4662/2-1 (535889065) and by the TU Dortmund - Center for Data Science & Simulation (DoDaS). *Simon Omlor*: supported by the German Research Foundation (DFG) – project no. 535889065. *Jeff M. Phillips*: thanks his support from NSF CCF-2115677, 2421782, and Simons Foundation MPS-AI-00010515.

Acknowledgements We would like to thank Peyman Afshani for valuable discussion.

1 Introduction

Linear Classification. One of the most fundamental tasks in machine learning is the minimum mislabeling linear classification problem. We formulate it as follows. Consider two point sets $R, B \subset \mathbb{R}^d$ (for red R and blue B points) so $|R \cup B| = n$. Let $\mathcal{H}_d$ be the set of all halfspaces parameterized by $w, \xi \in \mathbb{R}^d \times \mathbb{R}$. So the *halfspace* $h_{w,\xi} \in \mathcal{H}_d$ is the set of points $h_{w,\xi} := \{x \in \mathbb{R}^d \mid \langle w, x\rangle - \xi \geq 0\}$, where $\langle w, x\rangle = \sum_{j=1}^d w_j \cdot x_j$ denotes the standard Euclidean dot-product. Now given fixed input sets $R, B \subset \mathbb{R}^d$ and a halfspace $h \in \mathcal{H}_d$ define

$$\phi(h) = \phi_{R,B}(h) = |h \cap R| - |h \cap B|.$$

as the number of red points in h, minus the number of blue points in h. Then the fraction of misclassifications may be written as

$$\psi(h) = \frac{|R| - (|h \cap R| - |h \cap B|)}{|R \cup B|} = 1 - \frac{|B|}{|R \cup B|} - \frac{1}{n}\phi(h).$$

© Alexander Munteanu, Simon Omlor, and Jeff M. Phillips;
licensed under Creative Commons License CC-BY 4.0

So maximizing ϕ is equivalent to minimizing ψ. For historical reasons we write the aim to minimize the misclassification rate $\psi(h)$ as the following maximization problem and its approximate version:

- MaxHalfspace: For $R, B \subset \mathbb{R}^d$ with $|R \cup B| = n$, find $h^* = \arg\max_{h \in \mathcal{H}_d} \phi(h)$.
- ε-MaxHalfspace: Consider $R, B \subset \mathbb{R}^d$ with $|R \cup B| = n$ with $h^* = \arg\max_{h \in \mathcal{H}_d} \phi(h)$. For an error tolerance $\varepsilon \in (0, 1)$, find some $\hat{h} \in \mathcal{H}_d$ so that $\phi(h^*) - \phi(\hat{h}) \leq \varepsilon n$.

While it is common in machine learning to replace ψ with a convex loss function (such as logistic loss) which allows for a solution with gradient descent, the most classic formulations by Vapnik [57, 56], which begat much of the foundations of statistical learning theory, have this combinatorial form and result in strong generalization guarantees.

If $R \cup B$ is drawn i.i.d. from a distribution Π, then we can use $\psi(h)$ to predict how accurate h would be on a new data point $x \sim \Pi$. In particular, for $\varepsilon, \delta \in (0, 1)$, if $|R \cup B| = \Theta(\varepsilon^{-2}(d + \log(1/\delta)))$, then with probability at least $1 - \delta$ over the sample, we get a guarantee that the probability that any h correctly classifies a new point $x \sim \Pi$ is within ε to the finite sample fraction $\psi(h)$ [56, 43, 53]. Thus maximizing $\phi(h)$ on a small sample gives a classifier which is within ε to the best possible generalization guarantee for new data. Notably, this generalization rate guarantee is only accurate up to an additive $\varepsilon \in (0, 1)$. So solving for the exact optimum $h^* = \arg\max_{h \in \mathcal{H}_d} \phi(h)$ (as in MaxHalfspace) may be unnecessary. In such cases, it is sufficient to find some $\hat{h}$ satisfying ε-MaxHalfspace (and it can be randomized, succeeding with probability $1 - \delta$), to achieve the same asymptotic guarantees.

Conditional Hardness. Conditional hardness results in computational geometry are often obtained by reduction from base problems that are believed to be hard. The main problems we reduce from are k-Sum and AffineDegeneracy testing. We discuss key variants and known results starting from the classic 3-Sum [31].

Consider the k-Sum decision problem: Given $n \geq k$ integers $a_i \in \mathbb{Z}$, decide whether there exists a (multi)set $S \subseteq \{a_1, \ldots, a_n\}$ of $|S| = k$ integers that add up to 0.

The problem can be solved in time $O(n^{\lceil k/2 \rceil})$ by comparing $k/2$-tuples from one half of the input to the other half, known as the *meet in the middle* approach [39]. Despite polylogarithmic improvements for constant k [10], $\Omega(n^{\lceil k/2 \rceil})$ lower bounds hold against algorithms in certain restricted models of computation [27, 4] and in general the problem is conjectured to not admit $O(n^{\lceil k/2 \rceil - c})$ time algorithms for any $c > 0$, cf. [1].

▶ **Conjecture 1** (k-Sum conjecture). *For $k \geq 2$, there does not exist a randomized algorithm that succeeds (with high probability) in solving k-Sum in time $O(n^{\lceil k/2 \rceil - c})$ for any $c > 0$.*

Consider the AffineDegeneracy problem: Given n points in $\mathbb{Z}^d$, decide whether there exists a set of $(d + 1)$ points that lie on a common affine hyperplane in d dimensions.

If the algorithm is restricted to *sidedness* queries that return whether a given point lies above, below, or on a given affine hyperplane, then Erickson and Seidel [28] prove a $\Omega(n^d)$ lower bound matching the $O(n^d)$ complexity of any existing algorithm for the problem [15, 25, 24, 26]. Recent research by Cardinal and Sharir [12] has explored lower-order improvements in the exponent to $O(n^{k-1+o(1)})$ in the case of constant-degree polynomials in one dimension. Their result thus indicates that lifting the polynomial degree is slightly simpler than lifting to higher dimensions. The general case of AffineDegeneracy in d dimensions remains a major open problem in the real RAM model, for which it is conjectured that there exists no algorithm that runs in time $O(n^{d-c})$ for any $c > 0$, cf. [36, 13, 12].

▶ **Conjecture 2** (Affine Degeneracy conjecture). *There does not exist an algorithm that solves AffineDegeneracy in d dimensions in time $O(n^{d-c})$ for any $c > 0$.*

1.1 Our Results

We provide hardness results for exact and approximate versions of the MaxHalfspace problem that hold conditionally on the aforementioned widely believed hardness conjectures in the real RAM model of computation and unconditionally in a more restricted computational model that counts only $\text{sign}(x^T y) \in \{-1, 0, +1\}$ queries, referred to as *sidedness* query model or (ternary) k-linear decision tree model in the literature. For an overview, we only state our bounds slightly informally for the case d is a constant, and so for instance $2(d+1)^d$ is also a constant. The exact bounds are stated in the formal part in Section 2.

k-Sum Hardness. Our first reduction is from k-Sum to MaxDiscrepancy. This generalizes the previous 3-Sum reduction [31, 46] that gave quadratic lower bounds to an arbitrary $k \geq 2$ and hereby improves the exponents in the lower bounds to depend on the dimension. Our reduction yields the following theorem:

▶ **Theorem 3** (informal version of Theorem 6). *Assume that there exists no algorithm that solves k-Sum on n items within runtime $O(n^{\lceil k/2 \rceil - c})$ for any $c > 0$. Then there exist sets $R, B \subset \mathbb{R}^d, d = k - 1$, each of size $n/2$, for which there is no algorithm that solves either*

- MaxHalfspace *in time $O(n^{\lceil (d+1)/2 \rceil - c})$, or*
- ε-MaxHalfspace *in time $O(1/\varepsilon^{\lceil (d+1)/2 \rceil - c})$.*

Affine Degeneracy Testing Hardness. Our second reduction is from AffineDegeneracy testing to MaxDiscrepancy and yields the following theorems:

▶ **Theorem 4** (informal version of Theorem 8). *Assume that there exists no algorithm for AffineDegeneracy on n points with runtime $O(n^{d-c})$ for any $c > 0$. Then there exist sets $R, B \subset \mathbb{R}^d$, each of size $n/2$, for which there is no algorithm that solves either*

- MaxHalfspace *in time $O(n^{d-c})$, or*
- ε-MaxHalfspace *in time $O(1/\varepsilon^{d-c})$.*

▶ **Theorem 5.** *There exist sets $R, B \subset \mathbb{R}^d$, each of size $n/2$, for which any algorithm that uses only sidedness queries requires*

- $\Omega(n^d)$ *such queries to solve* MaxHalfspace, *and*
- $\Omega(1/\varepsilon^d)$ *such queries to solve* ε-MaxHalfspace.

We remark that lower bounds of the form $1/\varepsilon^d$ do not rule out $2^{1/\varepsilon}\,\text{poly}(d)$ algorithms. The latter two results are of special importance since either under the hardness conjecture or in the sidedness query model they match the best exact and approximate algorithms for this problem up to lower order polylogarithmic terms. The exact version, MaxHalfspace, has a $O(n^d)$ complexity [21, 22], and for the approximate version, ε-MaxHalfspace, there is an $O(\frac{1}{\varepsilon^d} \log^4 \frac{1}{\varepsilon})$ time algorithm (after preprocessing the input in $O(n)$ time) in [46]. Notably, this is based on a range counting data structure that can be implemented to work in the sidedness query model as well. The sidedness query model may seem restrictive at first glance and indeed it does not rule out simple comparisons between values (rather than signs) of dot products and determinants which may be exploited to break such lower bounds.

However, we stress that basically all known algorithms rely exactly on that type of subroutines, see related work below, and especially the discussion in [28]. Now, 30 years later, the core sidedness query operation is precisely the heavily optimized $\text{sign}(x^T y)$ operation which can be vectorized, optimized and accelerated on GPUs. Additionally, the study of big data models have brought up a paradigm that discretizes and reduces the search space to a

few candidate solutions and then brute forces that subset using simple queries as in [46]. This underlines that the sidedness query model is even more relevant in today's age of AI/ML and big data, than it probably was when Erickson and Seidel [28] studied it.

The hardness conjecture of AffineDegeneracy extends the same lower bound to the real RAM model in any dimension. In the Turing model, our reduction holds only in any constant dimension, due to the bit complexity of representing and adjusting small scalars.

1.1.1 Technical Overview

Reduction from k-Sum

Given an input $a_1, \ldots, a_n \in \mathbb{Z}$ for the k-Sum decision problem, the idea behind our construction is as follows: we construct for any $i \in [n]$ and $j \in [k]$ a point $x_{i,j} \in \mathbb{R}^{k-1}$. The points are constructed in such a way, that[1] the first coordinate of each $x_{i,j}$ corresponds to the input a_i, and for $j \in [1, \ldots, d-1]$, the remaining coordinates are the standard basis vectors e_j, whose coordinates are all 0 except for coordinate j which is 1. The remaining vectors are $x_{i,0}$ and $x_{i,d}$. Both have their first coordinate set to $-a_i$ (times a constant), then $x_{i,0}$ has all remaining coordinates set to 0, and $x_{i,d}$ has all remaining coordinates are set to 1.

The idea behind this construction is that a solution $a_1, \ldots, a_k$ to k-Sum corresponds to a hyperplane that crosses a specific selection of points, including an ancillary point that acts as an intercept term. Hence, there exists a linear combination of these points such that their first coordinates resemble the equation $a_1 + \ldots + a_{k-1} = -a_k$, while the other coordinates enforce constraints that ensure we make the right geometric selection of points, such that on these coordinates we have $e_1 + e_2 + \ldots + e_{k-1} = \mathbf{1}$, i.e., the selected unit basis vectors sum up to the vector $\mathbf{1}$ that comprises only 1s. By construction, a linear combination of these points whose coefficients parametrize a hyperplane also implies the existence of a k-Sum solution.

Now, we reduce to the MaxHalfspace problem. To this end, we copy each of the constructed points twice to create nk red points $r_{i,j} = x_{i,j} + \gamma e_1 \in R$, and another nk blue points $b_{i,j} = x_{i,j} - \gamma e_1 \in B$. That means the two colored point sets to be separated are simply copies of the previously constructed points but shifted by a tiny amount γ in opposite directions along their first coordinate. Note that this affects only the coordinate that corresponds to the original k-Sum input, not the additional gadgets.

Now if $||h \cap R| - |h \cap B|| \geq k$ then there exists a hyperplane that passes through the tiny gaps between at least k pairs of points $(r_{i,j}, b_{i,j})$. Using the constraints imposed by the gadgets this enforces a certain parameterization of the hyperplane. Applying this parameterization only to the first coordinates of the points and choosing $\gamma \leq 1/k$, we can bound $|a_1 + \ldots + a_k| \leq (k-1)\gamma < 1$. Since all a_i are integers, it follows that $a_1 + \ldots + a_k = 0$.

For the other direction $a_1 + \ldots + a_{k-1} = -a_k$ implies, as explained above, that there exists a linear combination of k original points $x_{i,j}$ that equals zero. Now note that the shifted versions $r_{i,j}, b_{i,j}$ of these k points each lie above (respectively below) the hyperplane, so each of them contributes 1 to the cost. Other points can only increase this number above k, or their red and blue contributions cancel in case they both lie on the same side. It thus follows that $||h \cap R| - |h \cap B|| \geq k$.

[1] up to technical scalars that are specified in the formal proof, see Section 2.1.

Reduction from AffineDegeneracy

This construction is more direct: given n points $x_i \in \mathbb{Z}^d$ as input for the AffineDegeneracy testing problem, we simply create n red points $r_i = x_i + \gamma e_1 \in R$, and another n blue points $b_i = x_i - \gamma e_1 \in B$. The two colored point sets to be separated are simply copies of the original points shifted by a tiny amount γ in opposite directions along their first coordinate.

If there exist $d+1$ points that lie on a common affine hyperplane, then the number of red points minus the number of blue points of the transformed input that lie in the halfspace h is clearly at least $||h \cap R| - |h \cap B|| \geq d+1$. To see this, note that for any of the original points on the hyperplane, we have one red point that lies above, and one blue point that lies below. Other points can only increase this number, or their red and blue contributions cancel if both lie on the same side.

The other direction is considerably more sophisticated and technical to prove: if $||h \cap R| - |h \cap B|| \geq d+1$ then there exist at least $d+1$ pairs (r_i, b_i) of corresponding red and blue points, such that the hyperplane passes through some points x'_i located in the tiny gap of size 2γ between r_i and b_i. Relabel these points $x'_1, \ldots, x'_{d+1}$. Our claim is that the corresponding $d+1$ original points $x_1, \ldots, x_{d+1}$ lie on a common affine hyperplane. Since any collection of d points such as $x_1, \ldots, x_d$ lie on a common hyperplane, the task reduces to show that x_{d+1} also lies on the same hyperplane.

Reframing this in terms of linear algebra, we collect the vectors $x'_1, \ldots, x'_d$ into a matrix $X' \in \mathbb{R}^{d \times d}$, and similarly $x_1, \ldots, x_d$ into $X \in \mathbb{Z}^{d \times d}$. Let $\mathbf{1} \in \mathbb{R}^d$ be the vector whose coordinates are all equal to one. We can now calculate the vector of coefficients of the hyperplane as $b' = X'^{-1}\mathbf{1}$. And for this instance, we know that $\langle b', x'_{d+1} \rangle - 1 = 0$. Similarly, we get for the original points $b = X^{-1}\mathbf{1}$, and our task is to prove that $\langle b, x_{d+1} \rangle - 1 = 0$ holds as well. To this end, we prove the following two claims:

(i) we have that $\langle b, x_{d+1} \rangle - 1$ is an integer multiple of $1/N_0$, for some integer $N_0 \neq 0$, and
(ii) it holds that

$$|\langle b, x_{d+1} \rangle - 1| = |\langle b, x_{d+1} \rangle - \langle b', x'_{d+1} \rangle + \langle b', x'_{d+1} \rangle - 1| = |\langle b, x_{d+1} \rangle - \langle b', x'_{d+1} \rangle| < 1/N_0.$$

Combining claims (i) and (ii) clearly implies our goal $\langle b, x_{d+1} \rangle - 1 = 0$.

At this point of the proof, especially bounding $|\langle b, x_{d+1} \rangle - \langle b', x'_{d+1} \rangle| < 1/N_0$ becomes very technical and requires bounding the effect of the remainder of matrix inverses $(X^{-1} - X'^{-1})$ along the shift direction e_1, as well as along x_{d+1}. Crucial steps involve showing that the determinants of X and its minors must be non-zero integers. Then we note that X' can be seen as a rank-one update of X, and apply the Sherman-Morrison formula to express the required remainder term in closed form. We further use Hadamard's bound on the determinants of minors of X in terms of the largest integer that appears in X. Choosing the shift size parameter γ small enough to cancel all but a factor $1/\det(X)$ yields the bounds that we require to conclude the proof. We refer the reader to Section 2.2 for the full details.

1.2 Implications Beyond Machine Learning

A number of other settings consider range space maximization, where the key unit of measure is the fraction of red points $\mu_R(h) = \frac{|h \cap R|}{|R|}$ and of blue points $\mu_B(h) = \frac{|h \cap B|}{|B|}$ in a halfspace.

- **discrepancy:** $\phi_{\|}(h) = |\mu_R(h) - \mu_B(h)|$
  This relates to discrepancy theory [49, 14, 9, 3] where one often seeks to find a coloring of a set $X \subset \mathbb{R}^d$ so each $x \in X$ is assigned to R or to B, and so $\max_{h \in \mathcal{H}} \phi_{\|}(h)$ is minimized over any choice of R, B. Then taking R, B as input, the optimization $\max_{h \in \mathcal{H}} \phi_{\|}(h)$

amounts to evaluating how well the chosen coloring works. In particular, many of the algorithms are Monte Carlo randomized, so to transform them into a Las Vegas style algorithm this verification step would be required. Moreover, iteratively applying discrepancy leads to coresets, which achieve additive $\varepsilon|X|$ error on ϕ or range counting queries; hence the approximate version is often sufficient.

In minimizing discrepancy, one often seeks $|R| = |B|$, in which case $\phi_{\|}(h) = \frac{2}{n}|\phi(h)|$. Further note that the absolute value $|\cdot|$ takes the maximum value between $\phi(h)$ and when reversing the roles of R and B. Our constructions will be symmetric in R and B with $|R| = |B|$. So our hardness results will hold for the most common case in discrepancy maximization in high dimensions.

- **Poisson:** $\phi_P(h) = \mu_R(h) \log \frac{\mu_R(h)}{\mu_B(h)} + (1 - \mu_R(h)) \log \frac{1-\mu_R(h)}{1-\mu_B(h)}$

  This relates to the spatial scan statistic model under a Poisson point process, as described by [42]. The maximizing $h \in \mathcal{H}$ corresponds to the most anomalous region under a statistical hypothesis test where the null hypothesis $\mathtt{H}_0$ models that there is a consistent Poisson process for all of the data, and the $\mathtt{H}_1$ hypothesis models that the Poisson rate is different in h than outside h. Then ϕ_P is the negative log-likelihood ratio of these two scenarios, also known as the KL divergence between the two distributions $(\mu_R(h), 1 - \mu_R(h))$ and $(\mu_B(h), 1 - \mu_B(h))$. The score $\phi_P(h^*)$ of the maximizing $h^* = \arg\max_{h \in \mathcal{H}} \phi_P(h)$ is the individually most powerful statistic for evaluating this sort of anomaly. Moreover, efficient ε-approximate algorithms [47] retain high statistical power. One can cover the relevant part of ϕ_P with $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ linear functions $\phi_\alpha(h) = \alpha|h \cap R| + (1 - \alpha)|h \cap B|$, so returning the maximum h over any function ε-approximately maximizes ϕ_P. And by changing the weight of points in R and B to α and $(1 - \alpha)$ our results provide hardness for any of these linear instances.

  In spatial scan statistics, it is common to search over balls [42, 47] instead of halfspaces (although halfspaces are considered [48]), and procedurally, a Veronese map from balls in $\mathbb{R}^{d-1}$ to halfspaces in $\mathbb{R}^d$ reduces this to a halfspace problem. All known methods with provable runtime and error guarantees [47, 46] either explicitly use or are isometric to using this transformation.

1.3 Additional Related Work

Maximum Discrepancy Classification. Exact algorithms are known that compute halfspace discrepancy in $O(n^d)$ time in any dimension $d \geq 2$ [21, 22, 23].

It was shown in [46] that the ε-MaxHalfspace problem can be solved in $O(\frac{1}{\varepsilon^d} \log^4 \frac{1}{\varepsilon})$ time (plus a linear scan over the data in time $O(n)$). This was complemented by conditional lower bounds in 2 dimensions: an exact solution requires $\Omega(n^{3/2})$ time conditioned on the conjecture that the all-pairs-shortest-paths problem (APSP) requires $\Omega(n^3)$ time to be solved. An approximate solution requires $\Omega(1/\varepsilon^2)$ by reduction from 3-Sum under the special case $k = 3$ of the aforementioned k-Sum conjecture detailed in Section 1.

Previous results by the same authors [45] gave $O(\frac{1}{\varepsilon^{d+1/3}} \log^{2/3} \frac{1}{\varepsilon})$ for ε-MaxHalfspace. They also studied a version that considered rectangular ranges instead of halfspaces, and provided an $O(n + \frac{1}{\varepsilon^2} \log\log \frac{1}{\varepsilon})$ algorithm and an $\Omega(1/\varepsilon^2)$ lower bound leveraging a previous $\Omega(n^2)$ lower bound of [6] for the exact rectangle variant. These lower bounds were again conditioned on the conjecture that APSP requires $\Omega(n^3)$ time to be solved.

More Connections to Learning Theory. Aside for the classic work described above, the hardness of classification has been studied in computational learning theory from other perspectives. A touchstone result by Guruswami and Raghvendra [34] showed it was NP-Hard in the dimension d to distinguish between instances where there exist halfspaces which

agree on at least a $(1-\varepsilon)$ fraction of the labels and ones where no halfspace agrees on more than a $(1/2+\xi)$ fraction of the data. Their work, and many others in this line (c.f. [30]), consider data from $\{0,1\}^d$, and since $O(d/\varepsilon^2)$ samples are sufficient to attain this ε-error, the number of points n does not show up separately in this analysis. But this work does not for instance control the constant in the exponent of the runtime; it does not rule out algorithms with runtime $1/\varepsilon^{\max\{2,d/1{,}000{,}000\}}$. Also, algorithms with runtime polynomial in $1/\varepsilon$ and d are possible with stronger noise like Massart noise [19].

A very recent and independent work by Pinto, Palit, and Raskhodnikova [52] approaches these problems from the property testing perspective, and conditioned on k-Sum, attains lower bound on the runtime for ε-MaxHalfspace of $\Omega(1/\varepsilon^{\lceil (d+1)/2\rceil - c})$. The core of their argument and construction is basically the same as our k-Sum hardness result.

No-Dimensional Results in Computational Geometry and Classification. Computational geometry has recently witnessed a number of "no-dimensional" or "dimension-free" results. These are often approximate solutions to problems in $\mathbb{R}^d$ where the core aspect of the problem does not depend on the dimension d. It may take $O(d)$ time to read or process each data point, so they are not independent of the dimension d, but importantly, these sorts of results prevent exponential dependence. Arguably this expedition started with approximate minimum enclosing ball and geometric median problems [8, 7, 40], which only require $O(1/\varepsilon)$ resp. $O(1/\varepsilon^2)$ points to approximate within a relative $(1+\varepsilon)$. This sort of analysis led to more general analyses of Frank-Wolfe style greedy algorithms [17], which induced no-dimensional results for the approximate polytope distance problem [33]. Also, classical combinatorial geometric consequences of convexity such as Carathéodory, Helly, and Tverberg theorems can be made no-dimensional when the diameter is bounded [2]. And this led to improved centerpoint and ε-net results [16, 37].

These geometric insights have led to improvements in sublinear and no-dimensional algorithms for classic linear classification problems [18, 20]. These analyses, and the above mentioned analysis of polytope distance [33], are refinements of Novikoff's classic analysis of the perceptron for linear classifiers, which is also no-dimensional. In particular, it requires that there exists a perfect classifier with no mistakes, and depends on the ratio between the distance between classes, and the diameter of the input set, cf. [38, 35]. The linear classification problem which optimizes an often convex loss function (e.g., logistic regression) requires additional assumptions to be made no-dimensional. Classic sparsity assumptions do not prevent from mild $\Theta(\log(d)/\varepsilon^2)$ dependence [44]. Strictly dimension-free bounds can be achieved if one uses a regularization term and also assumes the diameter is bounded (in expectation) [5]. These results $(1+\varepsilon)$-approximate typical classification loss functions with $O(1/\varepsilon^2)$ samples and an additional dependence that relies solely on the amount and type of regularization. Apart from classification there is a plethora of no-dimensional results in the clustering regime. The first dimension-free bounds for k-means/median clustering objectives appeared in [29, 54]. Recent developments extend the classic finite data setting to learning-theoretical dimension-free bounds for center based clustering and classification of points and polygonal curves [11, 41, 5].

Impossibility of No-Dimensional Results. In the context of these numerous no-dimensional results, it may be surprising that the geometric linear classification problem, and especially the approximately optimized variant tackled in this paper, is shown to require time exponential in d under conditional hardness assumptions. What is different in our setting? First, the perceptron-based results require separability, so $\psi(h^*)=0$; our constructions do not consider

this case, and have $\psi(h^*)$ fairly large. Second, these results mostly consider a bounded diameter of the point set, and also a structural term that prevents affine scaling, which could be the margin or the regularization term that has a similar effect. While we could affinely scale our constructions to be within a unit ball, it would make these other structural parameters (e.g., the margin between the properly separated points) proportionally small. Third, while no-dimensional ε-nets [37] and ε-covers [51] exist – and these play a crucial role in the best algorithms for our variant of classification [45, 46] – these results apply for bounded-radius balls or kernels, and not the halfspaces we study.

Hence, we hope our conditional hardness results help the community identify when these high-dimensional geometric problems relevant for data analysis can, *and cannot*, be solved in a dimension-efficient way. Our work provides recipes to show that even when approximation is allowed, exponential dependence on the dimension can be unavoidable.

2 Improved Halfspace Discrepancy Lower Bounds

2.1 k-Sum Hardness

We reduce from the k-Sum decision problem.

▶ **Theorem 6.** *Assume that there exists no algorithm that solves k-Sum on n items within runtime $O(n^{\lceil k/2\rceil - c})$ for any $c > 0$. Then there exist sets $R, B \subset \mathbb{R}^d$, for $d = k-1$, each of size $n/2$, for which there is no algorithm that solves either*

- MaxHalfspace *in time $O((n/(2(d+1))^{\lceil (d+1)/2\rceil - c})$, or*
- ε-MaxHalfspace *in time $O((1/(2(d+1)\varepsilon))^{\lceil (d+1)/2\rceil - c})$.*

Proof. We reduce k-Sum to MaxHalfspace with $d = k-1$ as follows. Let $A = \{a_1, \ldots, a_n\}$ for integers $a_i \in \mathbb{Z}$ be an instance of k-Sum. We will transform this input into a set of $2nk$ points in $\mathbb{R}^d$ where the solution to MaxHalfspace will resolve the k-Sum instance on A.

The Construction. For any integer $a_i, i \in [n]$ consider the following $k = d+1$ points in $\mathbb{R}^d$: Let $z_{i,0} := (-a_i/(2d-3), 0, \ldots, 0)$ and for each $j \in [d-1]$, let $z_{i,j} = (a_i/2, 0, \ldots, 0, 1, 0 \ldots, 0)$ which has a single 1 entry at position $j+1$. Further we create the point $z_{i,d} = (-a_i, 2, 2, \ldots, 2)$.

Further for each point $z_{i,j}$ we create two points: one point, which we place in R and one point, which we place in B. For $j \in \{0, \ldots, d\}$ we define $x_{i,j} \in R$ by $x_{i,j} = z_{i,j} + \gamma e_1$ and $y_{i,j} \in B$ by $y_{i,j} = z_{i,j} - \gamma e_1$ where γ is a sufficiently small number (for instance $\gamma < 1/(4d)$) and e_1 denotes the first standard basis vector in $\mathbb{R}^d$.

Restricting $R - B$ Difference to at least k. First, assume that there is h with $\mu_R|R| - \mu_B|B| \geq k$. Consider a set of $k = d+1$ lines defined by $l_0(t) = (t, 0, \ldots, 0)$, $l_j(t) = (t, 0, \ldots, 0, 1, 0, \ldots, 0)$, for $j \in [d-1]$, and $l_d(t) = (t, 2, \ldots, 2)$. Observe that all points in R and B lie on these lines. Let v_j be the point where h crosses the line l_j, for $j \in \{0, \ldots, d\}$.

Note, since the $z_{i,j}$ points are further apart than 2γ, then the number of points $x_{i,j}$ or $y_{i,j}$ on each of the lines that lie above h in R can differ by at most 1 from the number of points above h in B.

Thus, there cannot be any hyperplane with $\mu_R|R| - \mu_B|B| > k$ and in order to achieve $\mu_R|R| - \mu_B|B| = k$, the hyperplane h must cross each line l_j at a point $v_j = (\beta_{j+1}, \ldots)$ that for some $i_j \in [n]$, lies between the points $z_{i_j,j} - \gamma e_1$ and $z_{i_j,j} + \gamma e_1$. Equivalently, we have that $\beta_{j+1} \in [e_1^T z_{i,j} - \gamma, e_1^T z_{i,j} + \gamma]$. To simplify notation, we define $\alpha_{i,j} := e_1^T z_{i,j-1}$ for all $(i,j) \in [n] \times [k]$, where $k = d+1$. With this substitution, we get that $\beta_j \in [\alpha_{i,j} - \gamma, \alpha_{i,j} + \gamma]$.

Showing k-Separation implies k-Sum. Consider the points $w_j = v_j - v_0$, for $j \in [d]$. Since these are d points in $\mathbb{R}^d$, they must lie on a common linear hyperplane and the only way to align the upper $d-1$ coordinates is by setting $w_d = 2\sum_{j=1}^{d-1} w_j$.

This is equivalent to $0 = v_0 - v_d + 2\sum_{j=1}^{d-1}(v_j - v_0) = -(2d-3)v_0 + (2\sum_{j=1}^{d-1} v_j) - v_d$, implying in particular for the first coordinate that

$$-(2d-3)\beta_1 + \left(2\sum_{j=2}^{k-1}\beta_j\right) - \beta_k = 0. \tag{1}$$

Now, since for all $j \in [k]$ there exists some $i_j \in [n]$ such that $|\alpha_{i_j,j} - \beta_j| \le \gamma$, we get that

$$\begin{aligned}
&|a_{i_1} + \ldots + a_{i_k}| \\
&\overset{Eq.\ (1)}{=} \left| a_{i_1} + \ldots + a_{i_k} - \left(-(2d-3)\beta_1 + \left(2\sum_{j=2}^{k-1}\beta_j\right) - \beta_k\right)\right| \\
&= \left|-(2d-3)\alpha_{i_1,1} + \left(2\sum_{j=2}^{k-1}\alpha_{i_j,j}\right) - \alpha_{i_k,k} - \left(-(2d-3)\beta_1 + \left(2\sum_{j=2}^{k-1}\beta_j\right) - \beta_k\right)\right| \\
&\le (2d-3)|\alpha_{i_1,1} - \beta_1| + \left(\sum_{j=2}^{k-1} 2|\alpha_{i_j,j} - \beta_j|\right) + |\alpha_{i_k,k} - \beta_k| \le 4d\gamma < 1\,,
\end{aligned}$$

where we used the triangle inequality and the choice $\gamma < 1/(4d)$. Since all a_i are integers, it follows that $a_{i_1} + \ldots + a_{i_k} = 0$.

Showing k-Sum implies k-Separation. For the other direction, assume that $a_{i_1} + \ldots + a_{i_k} = 0$ or equivalently $a_{i_1} + \ldots + a_{i_{k-1}} = -a_{i_k}$. Then the points $z_{i_1,0}, \ldots, z_{i_k,d}$ lie on a hyperplane h. To prove this, note that any set of d points in d dimensions lie on a hyperplane. We need to show that the remaining point $z_{i_k,d}$ lies on the same hyperplane as the others. By a similar calculation as above, we get

$$\begin{aligned}
z_{i_k,d} = (-a_{i_k}, 2, \ldots, 2) &= (a_{i_1} + \ldots + a_{i_{k-1}}, 2, \ldots, 2) \\
&= -(2d-3)z_{i_1,0} + 2\sum_{j=1}^{d-1} z_{i_{j+1},j} = z_{i_1,0} + 2\sum_{j=1}^{d-1}(z_{i_{j+1},j} - z_{i_1,0}).
\end{aligned}$$

For this hyperplane it holds that $|\mu_R|R| - \mu_B|B|| \ge k$, since for any point $z_{i,l}$, for $l \in \{0, \ldots, d\}$ that lies on the hyperplane, the corresponding $x_{i,l}$ (and $y_{i,l}$) lie above (respectively below) the hyperplane, all counting for one of the two classes. If it does not lie on the hyperplane, then by construction and for sufficiently small γ as above, both $x_{i,l}$, and $y_{i,l}$ lie on the same side of the hyperplane and thus both contribute 0 or cancel each other.

Exact Bound. The number of points is $2nk = 2n(d+1)$, and so any algorithm with runtime $O((\beta n)^\alpha)$ for MaxHalfspace yields an algorithm with runtime $O((\beta n)^\alpha)$ for k-Sum. Set $\beta = 1/(2(d+1))$ and $\alpha = \lceil k/2 \rceil - c = \lceil (d+1)/2 \rceil - c$. Now using the Lemma's assumption on k-Sum hardness, there cannot be an algorithm for MaxHalfspace within runtime $O((\beta n)^\alpha) = O((n/(2(d+1))^{\lceil (d+1)/2 \rceil - c})$.

Cost of Reduction with Error ε. Similar to the reduction for MaxHalfspace, we now set $\varepsilon = 1/(2n(d+1))$, and any algorithm with runtime $O((\beta/\varepsilon)^\alpha)$ for ε-MaxHalfspace yields an algorithm with runtime $O((\beta/\varepsilon)^\alpha) = O(2n(d+1)\beta)^\alpha)$ for k-Sum. We again set $\beta = 1/(2(d+1))$ and $\alpha = \lceil k/2 \rceil - c = \lceil (d+1)/2 \rceil - c$. Thus, assuming the k-Sum hardness conjecture as in the statement of the lemma, there cannot be any algorithm that solves ε-MaxHalfspace within runtime $O(1/(2(d+1)\varepsilon)^{\lceil (d+1)/2 \rceil - c})$. We note that the time for the reduction from k-Sum to MaxHalfspace is $O(nk^2) \subseteq O(n^\alpha)$. ◀

2.2 Affine Degeneracy Hardness

The next lower bound is derived by reduction from the AffineDegeneracy problem. We first need a review of some basic linear algebra facts that are used in our proof.

2.2.1 Preliminaries and Review of Linear Algebra

Let $[d] = \{1, \ldots, d\}$ denote the positive integers up to d. Let $\mathbb{Z}_N = \{-N, \ldots, N\}$ be the set of integers between $-N$ and N. Given a square $d \times d$ matrix X, we denote its i,j-th entry for $i,j \in [d]$ by $X_{i,j}$ and its i-th row vector by X_i. Further, let its (i,j)-minor be the $(d-1) \times (d-1)$ submatrix denoted by $X^{\setminus(i,j)}$ that results from X by removing its i-th row and j-th column. We denote by I the $d \times d$ identity matrix and by e_i its i-th column vector.

Our reduction involves the analysis of the *determinant* $\det(X)$ and the *adjugate* matrix $\operatorname{adj}(X)$, whose entries are defined as $\operatorname{adj}(X)_{i,j} = (-1)^{i+j} \cdot \det(X^{\setminus(j,i)})$. We summarize certain well-known facts about these quantities and their relationships.

▶ **Fact.** *Let $X \in \mathbb{Z}_N^{d \times d}$ with linearly independent rows and columns. We remind of the following facts from linear algebra:*

1. $\det(X) \in \mathbb{Z}$
2. $1 \leq |\det(X)| \leq d^{d/2} N^d$
3. $X^{-1} = \det(X)^{-1} \operatorname{adj}(X)$
4. $\operatorname{adj}(X) \in \mathbb{Z}^{d \times d}$
5. *Let $u, v \in \mathbb{R}^d$. If X is invertible and $1 + u^T X^{-1} v \neq 0$, then*

$$X^{-1} - (X + u^T v)^{-1} = \frac{X^{-1} u v^T X^{-1}}{1 + u^T X^{-1} v}\,.$$

Proof. By Leibnitz' formula, also known as the *big formula* [55], the determinant of X can be written as a signed sum over products of the entries of X. Since each $X_{i,j} \in \mathbb{Z}_N \subset \mathbb{Z}$, these products are again in $\mathbb{Z}$ and signed sums of the products are again integers in $\mathbb{Z}$. We thus get Fact 1.

Next we have by linear independence, that $\det(X) \neq 0$. From this and Fact 1 it follows that $1 \leq |\det(X)|$. Hadamard's determinant bound [32] together with $X_{i,j} \in \mathbb{Z}_N$ yields that

$$1 \leq |\det(X)| \leq \prod_{j \in [d]} \left(\sum_{i \in [d]} X_{i,j}^2 \right)^{1/2} \leq \prod_{j \in [d]} (dN^2)^{1/2} = d^{d/2} N^d\,.$$

We thus conclude Fact 2.

Applying the Laplace expansion of the determinant, also known as *cofactor formula* [55], to every row of X it follows that $X \operatorname{adj}(X) = \det(X) I$. We recall that by linear independence, X is invertible. Left multiplication with X^{-1} thus yields $\operatorname{adj}(X) = X^{-1} X \operatorname{adj}(X) = X^{-1} \det(X) I$. By Fact 2, we have $\det(X) \neq 0$ and thus dividing both sides by $\det(X)$ concludes Fact 3, i.e., $X^{-1} = \det(X)^{-1} \operatorname{adj}(X)$.

Note that each (i,j)-minor of $X \in \mathbb{Z}_N^{d\times d}$ for $i,j \in [d]$ is a submatrix $X^{\setminus(i,j)} \in \mathbb{Z}_N^{(d-1)\times(d-1)}$. Then Fact 4 follows from Fact 1 since then $\operatorname{adj}(X)_{i,j} = (-1)^{i+j} \cdot \det(X^{\setminus(j,i)}) \in \mathbb{Z}$.

Fact 5 is a rearrangement of the Sherman-Morrison formula [55] to isolate the remainder term of the inverse under a rank-1 update $X^{-1} - (X + u^T v)^{-1}$. ◀

We will also need a bound on the coordinates of the inverse matrix multiplied by an arbitrary vector in the unit ℓ_∞ ball. To this end, we show that if X has integer coordinates bounded between $-N$ and N, then the coordinates in the following inverse term can be controlled in terms of N and d.

▶ **Lemma 7.** *Let $\lambda \in [-1,1]^d$ and $X \in \mathbb{Z}_N^{d\times d}$. Then for each coordinate $i \in [d]$ we have that*

$$|(X^{-1}\lambda)_i| \leq \frac{1}{|\det(X)|} \cdot d^{(d+1)/2} N^{d-1}.$$

Proof. By Hoelder's inequality, the fact that $\|\lambda\|_\infty \leq 1$, and using Hadamard's determinant bound (Fact 2), we have that

$$\begin{aligned}
|(X^{-1}\lambda)_i| &\overset{Fact\ 3}{=} |(\det(X)^{-1} \operatorname{adj}(X)\lambda)_i| = \frac{1}{|\det(X)|} \cdot |(\operatorname{adj}(X)\lambda)_i| \\
&\overset{Hoelder}{\leq} \frac{1}{|\det(X)|} \cdot \|\operatorname{adj}(X)_i\|_1 \|\lambda\|_\infty \leq \frac{1}{|\det(X)|} \cdot \sum_{j=1}^{d} \left|\det(X^{\setminus(j,i)})\right| \\
&\overset{Fact\ 2}{\leq} \frac{1}{|\det(X)|} \cdot d \cdot (d-1)^{(d-1)/2} N^{d-1} \leq \frac{1}{|\det(X)|} \cdot d^{(d+1)/2} N^{d-1}.
\end{aligned}$$

◀

2.2.2 Main Problem Reduction

We are now ready to prove the main result of this section.

▶ **Theorem 8.** *Assume that there exists no algorithm for* AffineDegeneracy *on n points with runtime $O(n^{d-c})$ for any $c > 0$. Then there exist sets $R, B \subset \mathbb{R}^d$, each of size $n/2$, for which there is no algorithm that solves either*

- MaxHalfspace *in time $O((n/2)^{d-c})$, or*
- ε-MaxHalfspace *in time $O(1/\varepsilon^{d-c})$.*

Proof. Let $X_n = \{x_1, \ldots, x_n\} \subset \mathbb{Z}_N^d$ be an instance of AffineDegeneracy. We construct an instance of MaxHalfspace by splitting each point into two points.

The Construction. For each $i \in [n]$ we create one point $y_i = x_i + \gamma e_1 \in R$ and one point $z_i = x_i - \gamma e_1 \in B$ where γ is sufficiently small (yet to be determined). Now, we claim that there exists a solution with $\mu_R|R| - \mu_B|B| \geq d+1$ if and only if there exists a set $X' \subseteq X_n$ of $|X'| = d+1$ points that lie on a common d-dimensional hyperplane.

First note that if there exists a set $X' \subseteq X_n$ of $|X'| = d+1$ points that lie on a common d-dimensional hyperplane h, then this hyperplane is a solution to MaxHalfspace with $\mu_R|R| - \mu_B|B| \geq d+1$ because for each $x \in X'$, the corresponding point $y \in R$ lies above h, whereas $z \in B$ lies below h.

For the other direction, assume that there exists a hyperplane h with $\mu_R|R| - \mu_B|B| \geq d+1$. Then there must be a set $S \subseteq [n]$ of $|S| = d+1$ indices such that for any $i \in S$ this hyperplane crosses $x_i' = x_i + \lambda_i \gamma e_1$ for some $\lambda_i \in [-1,1]$, i.e., it passes between $y_i \in R$ and $z_i \in B$ corresponding to x_i. Assume w.l.o.g. that $S = [d+1]$. It remains to show that $x_1, \ldots, x_{d+1}$ lie on a common hyperplane. The key will be leveraging that the coordinates of input points are integers to ensure that the solution to a satisfying halfspace must have x_{d+1} very close. Setting γ sufficiently small will imply the distance is actually 0.

Reframing the Problem in Linear Algebra. We assume w.l.o.g. that $x_1, \dots, x_d$ and $x'_1, \dots, x'_d$ are linearly independent, otherwise the proof continues verbatim in the largest dimension $d' < d$ for which the assumption is true, without affecting the hardness in d. Then $x'_1, \dots, x'_d$ lie on a common hyperplane and in particular there exists $b' \in \mathbb{R}^d$ such that for all $i \in [d]$, it holds that $\langle b', x'_i \rangle - 1 = 0$. We set $X \in \mathbb{Z}^{d \times d}$ to be matrix whose i-th row equals x_i^T, for $i \in [d]$, and $X' \in \mathbb{R}^{d \times d}$ to be matrix whose i-th row equals $x_i'^T$, for $i \in [d]$. Let $\mathbf{1} \in \mathbb{R}^d$ be the vector whose coordinates are all equal to one. Then we have that $X'b' = \mathbf{1}$, which implies $b' = X'^{-1}\mathbf{1}$. Similarly, we set $b = X^{-1}\mathbf{1}$.

Now, we show that $x_1, \dots, x_{d+1}$ lie on a common hyperplane by proving the following two claims, which clearly imply that $\langle b, x_{d+1} \rangle - 1 = 0$:

(i) we have that $\langle b, x_{d+1} \rangle - 1$ is a multiple of $1/N_0$, for some integer $N_0 \geq 1$, and

(ii) it holds that

$$|\langle b, x_{d+1} \rangle - 1| = |\langle b, x_{d+1} \rangle - \langle b', x'_{d+1} \rangle + \langle b', x'_{d+1} \rangle - 1| = |\langle b, x_{d+1} \rangle - \langle b', x'_{d+1} \rangle| < 1/N_0.$$

(i) Showing Integrality. For the first claim (i), note that $\langle b, x_{d+1} \rangle = (X^{-1}\mathbf{1})^T x_{d+1}$. Since X comprises only integer entries, $\det(X)$ is also an integer by Fact 1. By Fact 2, we also have that $|\det(X)| \geq 1$. We know from Fact 3 that $X^{-1} = \operatorname{adj}(X)/\det(X)$ and the entries of $\operatorname{adj}(X)$ are again integers by Fact 4. Finally, each coordinate $i \in [d]$ of $\operatorname{adj}(X)\mathbf{1}$ is a sum over the integer entries of $\operatorname{adj}(X)_i$. The vector x_{d+1} again consists only of integers. Taking their dot product thus retains the property of being integers, i.e., $(\operatorname{adj}(X)\mathbf{1})^T x_{d+1} \in \mathbb{Z}$.

We set $N_0 \coloneqq |\det(X)|$ and conclude that there exists $k \in \mathbb{Z}$ such that $\langle b, x_{d+1} \rangle - 1 = \frac{k}{N_0}$.

(ii) Controlling the Deviations. For the second claim (ii), let $R = X^{-1} - X'^{-1}$. To bound the remainder term R, we first express X' as a rank-1 update of X as follows. Let $u = \gamma\lambda$, and let $v = e_1$. Then $X'^{-1} = (X + uv^T)^{-1}$. Plugging in the definitions of u and v, the Sherman-Morrison formula yields

$$R = X^{-1} - X'^{-1} \overset{Fact\ 5}{=} \frac{X^{-1}uv^TX^{-1}}{1 + v^TX^{-1}u} = \frac{X^{-1}\lambda\gamma e_1^TX^{-1}}{1 + e_1^TX^{-1}\gamma\lambda} = \frac{X^{-1}\lambda\gamma e_1^TX^{-1}}{1 + (X^{-1}\lambda)^T\gamma e_1}\,.$$

We will see below that the denominator is non-zero, so the remainder term is well-defined. Now, recall that $X^{-1}\mathbf{1} = b$. Similarly, we set $\beta \coloneqq X^{-1}\lambda$. Using these substitutions, we get

$$R\mathbf{1} = (X^{-1} - X'^{-1})\mathbf{1} = \left(\frac{X^{-1}\lambda\gamma e_1^TX^{-1}\mathbf{1}}{1 + (X^{-1}\lambda)^T\gamma e_1}\right) = \left(\frac{\beta\gamma e_1^Tb}{1 + \beta^T\gamma e_1}\right) = \left(\frac{\gamma b_1}{1 + \gamma\beta_1}\right)\beta\,. \tag{2}$$

We choose $\gamma \coloneqq (6d^{d+2}N^{2d})^{-1} > 0$. In particular, with our coordinate bound (Lemma 7) this implies that $\gamma|\beta_1| \ll 1/2$, such that the denominator in the remainder term of Equation (2) is non-zero and bounded in the interval $[1/2, 3/2]$. By Lemma 7, we can bound the coordinates of β as $|\beta_i| = |(X^{-1}\lambda)_i| \leq \frac{1}{|\det(X)|} \cdot d^{(d+1)/2}N^{d-1}$. Since $\mathbf{1} \in [-1, 1]^d$ the same bound holds for all coordinates $|b_i|$ of b. These arguments yield the following bound

$$\begin{aligned}
\left|(R\mathbf{1})^T x_{d+1}\right| &\overset{Eq.\ (2)}{=} \left|\left(\frac{\gamma b_1}{1 + \gamma\beta_1}\right)\beta^T x_{d+1}\right| \overset{Hoelder}{\leq} \gamma\left|\left(\frac{1}{1 + \gamma\beta_1}\right)\right| |b_1| \|\beta\|_\infty \|x_{d+1}\|_1 \\
&\overset{Lem.\ 7}{\leq} \gamma\left|\left(\frac{1}{1 + \gamma\beta_1}\right)\right| \cdot \frac{1}{|\det(X)|^2} \cdot d^{d+1}N^{2d-2} \cdot dN \\
&< \quad 2\gamma \cdot \frac{1}{|\det(X)|} \cdot d^{d+2}N^{2d-1} < \frac{1}{3 \cdot |\det(X)|}\,.
\end{aligned} \tag{3}$$

With almost the same calculation, we also have that

$$\begin{aligned}\left|(R\mathbf{1})^T e_1\right| &\overset{Eq.\ (2)}{=} \left|\left(\frac{\gamma b_1}{1+\gamma\beta_1}\right)\beta_1\right| \leq \gamma\left|\left(\frac{1}{1+\gamma\beta_1}\right)\right||b_1||\beta_1| \\ &\overset{Lem.\ 7}{\leq} \gamma\left|\left(\frac{1}{1+\gamma\beta_1}\right)\right| \cdot \frac{1}{|\det(X)|^2}\cdot d^{d+1}N^{2d-2} \\ &< \quad 2\gamma\cdot\frac{1}{|\det(X)|}\cdot d^{d+1}N^{2d-2} < \frac{1}{3\cdot|\det(X)|}\,. \end{aligned} \tag{4}$$

Now, by combining the triangle inequality, Equation (4), Lemma 7, and our choice of $\gamma = (6d^{d+2}N^{2d})^{-1} \leq 1$, we get that

$$\begin{aligned}\gamma\left|(R\mathbf{1})^T e_1 - (X^{-1}\mathbf{1})^T e_1\right| &\leq \gamma|(R\mathbf{1})^T e_1| + \gamma|b_1| \\ &< \frac{1}{3\cdot|\det(X)|} + \frac{1}{6\cdot|\det(X)|} = \frac{1}{2\cdot|\det(X)|}\,. \end{aligned} \tag{5}$$

We conclude the second claim (ii) as follows

$$\begin{aligned}&|\langle b, x_{d+1}\rangle - \langle b', x'_{d+1}\rangle| \\ &= |(X^{-1}\mathbf{1})^T x_{d+1} - (X'^{-1}\mathbf{1})^T x'_{d+1}| \\ &= |(X^{-1}\mathbf{1})^T x_{d+1} - (X'^{-1}\mathbf{1} - X^{-1}\mathbf{1} + X^{-1}\mathbf{1})^T x'_{d+1}| \\ &= |(X^{-1}\mathbf{1})^T x_{d+1} - (X^{-1}\mathbf{1})^T x'_{d+1} + (X^{-1}\mathbf{1} - X'^{-1}\mathbf{1})^T x'_{d+1}| \\ &= |(X^{-1}\mathbf{1})^T x_{d+1} - (X^{-1}\mathbf{1})^T x_{d+1} - \gamma\lambda_{d+1}(X^{-1}\mathbf{1})^T e_1 + (R\mathbf{1})^T x'_{d+1}| \\ &= |(R\mathbf{1})^T x_{d+1} + \gamma\lambda_{d+1}(R\mathbf{1})^T e_1 - \gamma\lambda_{d+1}(X^{-1}\mathbf{1})^T e_1| \\ &\leq |(R\mathbf{1})^T x_{d+1}| + \gamma|\lambda_{d+1}||(R\mathbf{1})^T e_1 - (X^{-1}\mathbf{1})^T e_1| \\ &< \frac{1}{3\cdot|\det(X)|} + \frac{1}{2\cdot|\det(X)|} < \frac{1}{|\det(X)|}\,, \end{aligned}$$

where the triangle inequality is followed by using the fact that $|\lambda_{d+1}| \leq 1$ and applying Equations (3) and (5).

Final Reductions. Finally, we set $m = n/2$ and $\varepsilon = 1/n$. Then, any algorithm that solves MaxHalfspace on R and B in time $O(m^{d-c}) = O((n/2)^{d-c})$ time or ε-MaxHalfspace in $O(\varepsilon^{-(d-c)})$ time can also be used to solve AffineDegeneracy in time $O(n^{d-c})$. ◀

References

1 Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-sum conjecture. In *Automata, Languages, and Programming - 40th International Colloquium (ICALP)*, pages 1–12. Springer, 2013. doi:10.1007/978-3-642-39206-1_1.

2 Karim Adiprasito, Imre Bárány, Nabil H Mustafa, and Tamás Terpai. Theorems of Carathéodory, Helly, and Tverberg without dimension. *Discrete & Computational Geometry*, 64(2):233–258, 2020. doi:10.1007/S00454-020-00172-5.

3 Manasseh Ahmed, Tsun-Ming Cheung, Hamed Hatami, and Kusha Sareen. Communication Complexity and Discrepancy of Halfplanes. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.5.

4 Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005. doi:10.1145/1059513.1059515.

5 Meysam Alishahi and Jeff M. Phillips. No dimensional sampling coresets for classification. In *Forty-first International Conference on Machine Learning, (ICML)*, pages 1008–1049, 2024. URL: https://openreview.net/forum?id=jS3CMHtYJD.

6 Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 81:1–81:13, 2016. doi:10.4230/LIPIcs.ICALP.2016.81.

7 Mihai Bădoiu and Kenneth L. Clarkson. Smaller core-sets for balls. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 801–802, 2003.

8 Mihai Bădoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, pages 250–257, 2002.

9 Nikhil Bansal. Constructive algorithms for discrepancy minimization. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 3–10, 2010. doi:10.1109/FOCS.2010.7.

10 Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008. doi:10.1007/S00453-007-9036-3.

11 Maria Sofia Bucarelli, Matilde Fjeldsø Larsen, Chris Schwiegelshohn, and Mads Toftrup. On generalization bounds for projective clustering. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/e30bf4765ae6b16a87fb4d7b0b3b3dec-Abstract-Conference.html.

12 Jean Cardinal and Micha Sharir. Improved algebraic degeneracy testing. *Discret. Comput. Geom.*, 74(1):177–195, 2025. doi:10.1007/S00454-024-00673-7.

13 Timothy M Chan and Da Wei Zheng. Hopcroft's problem, log-star shaving, 2d fractional cascading, and decision trees. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 190–210, 2022. doi:10.1137/1.9781611977073.10.

14 Bernard Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, 2000.

15 Bernard Chazelle, Leonidas J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25(1):76–90, 1985. doi:10.1007/BF01934990.

16 Aruni Choudhary and Wolfgang Mulzer. No-dimensional Tverberg theorems and algorithms. *Discrete & Computational Geometry*, 68(4):964–996, 2022. doi:10.1007/S00454-022-00380-1.

17 Kenneth L Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):1–30, 2010. doi:10.1145/1824777.1824783.

18 Kenneth L Clarkson, Elad Hazan, and David P Woodruff. Sublinear optimization for machine learning. *Journal of the ACM (JACM)*, 59(5):1–49, 2012. doi:10.1145/2371656.2371658.

19 Ilias Diakonikolas, Themis Gouleakis, and Christos Tzamos. Distribution-independent pac learning of halfspaces with massart noise. *Advances in Neural Information Processing Systems*, 32, 2019.

20 Hu Ding. Stability Yields Sublinear Time Algorithms for Geometric Optimization in Machine Learning. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms (ESA 2021)*, volume 204 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2021.38.

21 David P. Dobkin and David Eppstein. Computing the discrepancy. In *Proceedings of the Ninth Annual Symposium on Computational Geometry (SoCG)*, pages 47–52, 1993. doi:10.1145/160985.160997.

22 David P. Dobkin, David Eppstein, and Don P. Mitchell. Computing the discrepancy with applications to supersampling patterns. *ACM Trans. Graph.*, 15(4):354–376, 1996. doi:10.1145/234535.234536.

23 David P Dobkin, Dimitrios Gunopulos, and Wolfgang Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *Journal of Computer and System Sciences*, 52(3):453–470, 1996. doi:10.1006/JCSS.1996.0034.

24 Herbert Edelsbrunner and Leonidas J. Guibas. Topologically sweeping an arrangement. *J. Comput. Syst. Sci.*, 38(1):165–194, 1989. doi:10.1016/0022-0000(89)90038-X.

25 Herbert Edelsbrunner, Joseph O'Rourke, and Raimund Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15(2):341–363, 1986. doi:10.1137/0215024.

26 Herbert Edelsbrunner, Raimund Seidel, and Micha Sharir. On the zone theorem for hyperplane arrangements. *SIAM J. Comput.*, 22(2):418–429, 1993. doi:10.1137/0222031.

27 Jeff Erickson. Lower bounds for linear satisfiability problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 388–395, 1995. URL: http://dl.acm.org/citation.cfm?id=313651.313772.

28 Jeff Erickson and Raimund Seidel. Better lower bounds on detecting affine and spherical degeneracies. *Discret. Comput. Geom.*, 13:41–57, 1995. doi:10.1007/BF02574027.

29 Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning Big Data into tiny data: Constant-size coresets for k-means, PCA, and projective clustering. *SIAM J. Comput.*, 49(3):601–657, 2020. doi:10.1137/18M1209854.

30 Vitaly Feldman, Venkatesan Guruswami, Prasad Raghavendra, and Yi Wu. Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6):1558–1590, 2012. doi:10.1137/120865094.

31 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995. doi:10.1016/0925-7721(95)00022-2.

32 D. J. H. Garling. *Inequalities: A Journey into Linear Analysis*. Cambridge University Press, 2007.

33 Bernd Gärtner and Martin Jaggi. Coresets for polytope distance. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 33–42, 2009. doi:10.1145/1542362.1542370.

34 Venkatesan Guruswami and Prasad Raghavendra. Hardness of learning halfspaces with noise. *SIAM Journal on Computing*, 39(2):742–765, 2009. doi:10.1137/070685798.

35 Sariel Har-Peled. A simple algorithm for maximum margin classification, revisited. *CoRR*, abs/1507.01563, 2015. arXiv:1507.01563.

36 Sariel Har-Peled, Piotr Indyk, and Sepideh Mahabadi. Approximate sparse linear regression. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 77:1–77:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.77.

37 Sariel Har-Peled and Eliot W Robson. No-dimensional Tverberg partitions revisited. *Discrete & Computational Geometry*, pages 1–19, 2025.

38 Sariel Har-Peled, Dan Roth, and Dav Zimak. Maximum margin coresets for active and noise tolerant learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 836–841, 2007. URL: http://ijcai.org/Proceedings/07/Papers/134.pdf.

39 Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM (JACM)*, 21(2):277–292, 1974. doi:10.1145/321812.321823.

40 Amer Krivosija and Alexander Munteanu. Probabilistic smallest enclosing ball in high dimensions via subgradient sampling. In *35th International Symposium on Computational Geometry (SoCG)*, pages 47:1–47:14, 2019. doi:10.4230/LIPIcs.SOCG.2019.47.

41 Amer Krivošija, Alexander Munteanu, André Nusser, and Chris Schwiegelshohn. Improved learning via k-DTW: A novel dissimilarity measure for curves. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, pages 31696–31726, 2025. URL: https://proceedings.mlr.press/v267/krivosija25a.html.

42 Martin Kulldorff. A spatial scan statistic. *Communications in Statistics-Theory and methods*, 26(6):1481–1496, 1997.

43 Yi Li, Philip M Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62(3):516–527, 2001. doi:10.1006/JCSS.2000.1741.

44 Tung Mai, Alexander Munteanu, Cameron Musco, Anup Rao, Chris Schwiegelshohn, and David P. Woodruff. Optimal sketching bounds for sparse linear regression. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 11288–11316, 2023. URL: https://proceedings.mlr.press/v206/mai23a.html.

45 Michael Matheny and Jeff M. Phillips. Computing approximate statistical discrepancy. In *29th International Symposium on Algorithms and Computation (ISAAC)*, pages 32:1–32:13, 2018. doi:10.4230/LIPIcs.ISAAC.2018.32.

46 Michael Matheny and Jeff M. Phillips. Approximate maximum halfspace discrepancy. In *32nd International Symposium on Algorithms and Computation (ISAAC)*, pages 4:1–4:15, 2021. doi:10.4230/LIPIcs.ISAAC.2021.4.

47 Michael Matheny, Raghvendra Singh, Liang Zhang, Kaiqiang Wang, and Jeff M Phillips. Scalable spatial scan statistics through sampling. In *Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 1–10, 2016. doi:10.1145/2996913.2996939.

48 Michael Matheny, Dong Xie, and Jeff M Phillips. Scalable spatial scan statistics for trajectories. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(6):1–24, 2020. doi:10.1145/3394046.

49 Jiri Matousek. *Geometric discrepancy: An illustrated guide*, volume 18. Springer Science & Business Media, 1999.

50 Alexander Munteanu, Simon Omlor, and Jeff M. Phillips. Hardness of high-dimensional linear classification, 2026. arXiv:2603.19061.

51 Jeff Phillips and Hasan Pourmahmoodaghababa. Dimension-independent kernel eps-covers. *Computing in Geometry and Topology*, 4(1):5–1, 2025.

52 Renato Ferreira Pinto Jr, Diptaksho Palit, and Sofya Raskhodnikova. Computational complexity in property testing. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 408–440. SIAM, 2026.

53 Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014.

54 Christian Sohler and David P. Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. In *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 802–813, 2018. doi:10.1109/FOCS.2018.00081.

55 Gilbert Strang. *Introduction to Linear Algebra*. Cambridge University Press, 5 edition, 2021.

56 Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity: Festschrift for Alexey Chervonenkis*, pages 11–30. Springer, 2015.

57 V.N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999. doi:10.1109/72.788640.

Optimal-Cost Construction of Shallow Cuttings for 3-D Dominance Ranges in the I/O-Model

Yakov Nekrich
Michigan Technological University, Houghton, MI, USA

Saladi Rahul
Indian Institute of Science, Bangalore, India

Abstract

Shallow cuttings are a fundamental tool in computational geometry and spatial databases for solving offline and online range searching problems. For a set P of N points in 3-D, at SODA'14, Afshani and Tsakalidis designed an optimal $O(N \log_2 N)$ time algorithm that constructs shallow cuttings for 3-D dominance ranges in internal memory. Even though shallow cuttings are used in the I/O-model to design space and query efficient range searching data structures, an efficient construction of them is not known till now. In this paper, we design an optimal-cost algorithm to construct shallow cuttings for 3-D dominance ranges. The number of I/Os performed by the algorithm is $O\left(\frac{N}{B} \log_{M/B}\left(\frac{N}{B}\right)\right)$, where B is the block size and M is the memory size.

As two applications of the optimal-cost construction algorithm, we design fast algorithms for offline 3-D dominance reporting and offline 3-D approximate dominance counting. We believe that our algorithm will find further applications in offline 3-D range searching problems and in improving construction cost of data structures for 3-D range searching problems.

2012 ACM Subject Classification Theory of computation → Data structures and algorithms for data management; Theory of computation → Computational geometry

Keywords and phrases Data Structures, I/O-efficient algorithms, Orthogonal Range Searching

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.81

Related Version *Full Version*: `https://arxiv.org/abs/2603.21337`

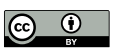
Funding *Yakov Nekrich*: Supported by the National Science Foundation under NSF grant 2203278. *Saladi Rahul*: Supported by the Walmart Center for Tech Excellence at IISc (CSR Grant WMGT-230001) and Anusandhan National Research Foundation (SP/ANRF-24-0043 CRG/2023/005776).

Acknowledgements The authors thank the reviewers whose detailed feedback helped in improving the presentation of the paper.

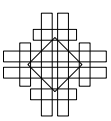
1 Introduction

Shallow cuttings are one of the most fundamental tools used for designing range searching data structures in computational geometry [1, 2, 3, 4, 13] and spatial databases [16, 17, 6]. The main contribution of this paper is the first known optimal-cost construction of shallow cuttings for *3-D dominance ranges* in the *I/O-model*. As a consequence, we obtain efficient algorithms for the *offline 3-D dominance reporting* problem and the *offline 3-D approximate counting* problem.

Shallow cuttings for 3-D dominance ranges. Consider two 3-D points p and q. A point p *dominates* another point q if and only if p has a higher coordinate value than q in each dimension. Let P be a set of N points in 3-D. A *k-level shallow cutting* of P [1] (See Figure 1(a)) is a collection $\mathcal{C}$ of *cells* of the form $(-\infty, x] \times (-\infty, y] \times (-\infty, z]$ satisfying the following three properties:



© Yakov Nekrich and Saladi Rahul;
licensed under Creative Commons License CC-BY 4.0



1. The number of cells is $O(n/k)$, i.e., $|\mathcal{C}| = O(n/k)$.
2. Each cell in $\mathcal{C}$ contains at most $10k$ points from P.
3. Any 3-D point that dominates at most k points of P lies inside at least one cell in $\mathcal{C}$.

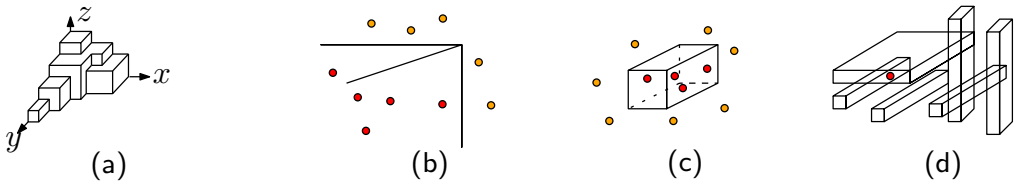


Figure 1 (a) A schematic representation of a k-level 3-D shallow cutting, (b) 3-D dominance reporting and approximate counting query, (c) 3-D orthogonal range reporting query, (d) 3-D rectangle stabbing query.

I/O-model of computation. We will construct shallow cuttings for 3-D dominance ranges in the I/O-model of computation [21]. In the era of big-data, it is natural that the input will not fit completely in the internal memory, and hence, resides in the hard-disk (or the external memory). In this model, the internal memory has M words, and the hard-disk has been formatted into *blocks* of B words. An I/O operation exchanges a block of data between the disk and the memory. The space of a data structure is the number of blocks occupied in the external memory, and the running time of an algorithm is the number of I/Os performed. The CPU calculation is free. A natural assumption is that $M = \Omega(B)$.

3-D orthogonal range searching. Orthogonal range searching and related problems are foundational problems on which the fields of computational geometry and spatial databases grew. Naturally, they have been well studied in the I/O-model of computation. Consider the following problems for which the state-of-the-art data structures in the I/O-model extensively rely on the clever use of shallow cuttings for 3-D dominance ranges:

1. In the *3-D orthogonal range reporting* problem (Figure 1(c)), the input is a set P of N points in 3-D. Preprocess P into a space-efficient data structure, so that given an axis-aligned query box $q = [x_1, x_2] \times [y_1, y_2] \times [z_1, z_2]$, the goal is to quickly report $P \cap q$ (the points of P lying inside q). AT FOCS'09, Afshani, Arge, and Larsen [2] presented a data structure for this problem with space $O\left(\frac{N}{B}\left(\frac{\log N}{\log\log_B N}\right)^3\right)$ and query I/Os $O(\log_B N + k/B)$, where k is the number of points reported.
2. The building block for the above problem is the *3-D dominance reporting* problem (Figure 1(b)), where the query is a 3-D dominance range $q = (-\infty, x] \times (-\infty, y] \times (-\infty, z]$. At ESA'08, Afshani [14] presented a data structure for this problem with space $O(N/B)$ and query I/Os $O(\log_B N + k/B)$. This is optimal in terms of space and query I/Os.
3. At SoCG'09, Afshani, Hamilton, and Zeh presented a general approach to design data structures to answer the *relative* $(1+\varepsilon)$*-approximate counting* query for a general class of problems including the 3-D dominance setting. Here the goal is to output the estimate of $|P \cap q|$. An optimal solution in terms of space and query I/Os was obtained.
4. *Rectangle stabbing reporting* (Figure 1(d)) is the "reverse" of orthogonal range reporting where the input is a set of axis-aligned 3-D boxes and the query is a 3-D point. Arguably, rectangle stabbing reporting is as important as orthogonal range reporting and is well studied in the literature. The current state-of-the-art results in internal memory models of computation for rectangle stabbing reporting by Rahul [15] at SODA'15 and Chan,

Nekrich, Rahul, and Tsakalidis [9] at ICALP'18 rely on shallow cuttings for 3-D dominance ranges. It is unlikely that a linear-space data structure in the I/O-model can be constructed without using shallow cuttings.

Lack of fast construction of data structures. An important criteria in the design of the data structures is to optimize the I/Os required to construct the data structure. A severe limitation of *all* the above solutions is the lack of such fast construction algorithms. In fact, quoting from one of the above papers (Afshani, Hamilton, and Zeh [4]),

"All data structures presented in this paper are static, and no efficient construction methods for these structures are known in the I/O or cache-oblivious model. The main obstacle is the lack of an I/O-efficient or cache-oblivious construction procedure for shallow cuttings."

We believe that our work on optimal-cost construction of shallow cuttings for 3-D dominance ranges will trigger further work to design fast construction of data structures for the above mentioned problems.

Lack of optimal solutions for offline 3-D orthogonal range searching. In the *offline* orthogonal range searching problem, we are given a set Q of queries upfront along with the point set P. Many 2-D offline problems in computational geometry (including range searching type problems) have been optimally solved in the I/O-model. See the survey paper by Vitter (Theorem 8.1 in [21] provides a comprehensive list of more than ten problems in 2-D which are solved I/O-optimally).

However, to the best of our knowledge, offline 3-D orthogonal range searching problems have not received the same attention in spite of their fundamental nature. Again, the lack of optimal-cost shallow cuttings in 3-D is the major bottleneck.

1.1 Our results

The main result in this paper is an optimal-cost construction of the k-level shallow cutting for 3-D dominance ranges in the I/O-model.

▶ **Theorem 1.** *The k-level shallow cutting for 3-D dominance ranges on N points can be constructed in $O\left(\frac{N}{B}\log_{M/B}\left(\frac{N}{B}\right)\right)$ I/Os. The construction cost is optimal in the I/O-model.*

To demonstrate the impact of our new construction, we use shallow cuttings for 3-D dominance ranges to obtain efficient algorithms for two fundamental problems: *offline 3-D dominance approximate counting* and *offline 3-D dominance reporting*.

▶ **Theorem 2** (Application-1)**.** *Let P be a set of N points in 3-D and Q be a set of 3-D dominance ranges. In the offline 3-D dominance approximate counting problem, for any $q \in Q$, the goal is to return a value in the range $[(1-\varepsilon)|P \cap q|, |P \cap q|]$, where $\varepsilon \in (0,1)$ is a fixed value. There is a deterministic algorithm to solve the problem in $O_\varepsilon\left(\frac{|P|}{B}\left(\log_{M/B}\frac{|P|}{B}\right)^{O(1)} + \frac{|Q|}{B}\log_{M/B}\frac{|P|}{B}\right)$ I/Os. The notation $O_\varepsilon(\cdot)$ hides the dependency on ε.*

The key feature in the above theorem is that the construction cost bound has the term M/B in the base of the logarithm which is desirable in the I/O-model (instead of *base*-2 obtained from straightforward adaptation of internal memory algorithms or *base-B* obtained from straightforward adaption of external memory data structures for online problems). In

terms of techniques, even though the final output is a relative approximation, the crucial ingredients in the algorithm are an additive error counting structure that is combined with a non-trivial *van-Emde-Boas-style recursion.*

▶ **Theorem 3** (Application-2). *Let P be a set of N points in 3-D and Q be a set of 3-D dominance ranges. In the offline 3-D dominance reporting problem, for any $q \in Q$, the goal is to report $P \cap q$. The problem can be solved in $O\left(\frac{|P|+|Q|}{B} \log_{M/B} \frac{|P|}{B} + \frac{K}{B}\right)$ I/Os, where K is the output size.*

The final algorithm is a combination of several ideas: filtering search (to charge the I/Os performed for a query to its output-size), optimal k-level shallow cuttings (which were not known before), roughly $O(\log \log N)$ levels of shallow cuttings, and a cleverly chosen base-case. The non-trivial part of the analysis is bounding the number of I/Os to perform the so-called offline-find-any procedures (defined later).

To prove the above theorems, we make use of data structures with specific requirements. For example, in Theorem 4 data structures with fast construction I/Os are considered and in Theorem 5 a data structure with large additive error is considered. Although there exist data structures with desired bounds to handle some of these queries [11, 12, 18, 19], we are not aware of any prior work that obtains these results using a common framework, which might be useful for other problems.

▶ **Theorem 4** (Reporting, counting, and selection). *Let P be a set of N points in 3-D and γ be a fixed constant in $(0, 1)$. Then there exists a data structure that can be constructed in $O(sort(N)) = O\left(\frac{N}{B} \log_{M/B}\left(\frac{N}{B}\right)\right)$ I/Os and then used to answer a*

- *3-D dominance reporting query in $O((N/B)^\gamma + t/B)$ I/Os, where t is the number of points reported. Given a 3-D dominance query range q, the goal is to report the points in $P \cap q$.*
- *3-D dominance counting query in $O((N/B)^\gamma)$ I/Os. Given a 3-D dominance query range q, the goal is to report $|P \cap q|$.*
- *3-D dominance x, y, z-selection query in $O((N/B)^\gamma)$ I/Os. In an x-selection query, given a query point $q = (q_x, q_y, q_z)$ and an integer $k' \in [1, N]$, the goal is to return a point $q_1 = (q'_x, q_y, q_z)$ that dominates k' points in P. Analogously, in a y-selection query (resp., z-selection query), the goal is to return a point $q_2 = (q_x, q'_y, q_z)$ (resp., $q_3 = (q_x, q_y, q'_z)$) that dominates k' points in P.*

▶ **Theorem 5** (Offline additive error counting). *Let P be a set of N points in 3-D and Q be a set of 3-D dominance ranges. In the offline additive error counting problem, for each query $q \in Q$, the goal is to return a value n_q such that $|P \cap q| - n_q \leq \varepsilon N^{2/3} B^{1/3}$. The data structure can be constructed using $O_\varepsilon(sort(N))$ I/Os and the queries are answered by using $O_\varepsilon\left(sort(N) + sort(|Q|)\right)$ I/Os.*

Our solution for additive error query can be extended to handle smaller values of error (such as $\varepsilon\sqrt{NB}$). However, for our purpose an additive error of $\varepsilon N^{2/3} B^{1/3}$ suffices.

The remaining paper is structured as follows. In Section 2, we discuss some preliminaries. In Section 3, we give an overview of two previous constructions of shallow cuttings for 3-D dominance ranges in internal memory. Then, in Section 4 we present our optimal-cost construction in the I/O-model. This is followed by two applications of our construction, offline 3-D dominance approximate counting (in Section 5) and offline 3-D dominance reporting (in the full version). In the full version, we also present the details of our supporting structures (Theorem 4 and Theorem 5) and potential directions for future work.

2 Preliminaries

Definitions. The *conflict list* of a cell $C \in \mathcal{C}$ is defined as the points of P that are inside C. The *apex point* of a cell or a 3-D dominance range $(-\infty, x] \times (-\infty, y] \times (-\infty, z]$ is defined as (x, y, z). The *depth* of a point $p = (x_p, y_p, z_p)$ w.r.t. a pointset P is the number of points of P lying in the cell $(-\infty, x_p] \times (-\infty, y_p] \times (-\infty, z_p]$.

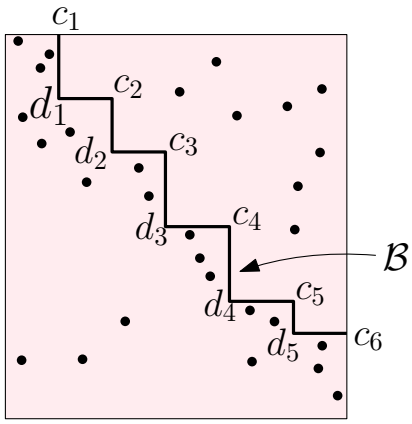


Figure 2 An example of a k-level 2-D shallow cutting. Here $k = 3$, so each inner corner d_i dominates three points and each outer corner c_i dominates six points.

Shallow cuttings for 2-D dominance ranges. As a warm-up, we first introduce shallow cuttings for *2-D dominance ranges*. Let P be a set of N points in 2-D. The collection $\mathcal{C}$ of cells in the k-level shallow cutting of P will be of form $(-\infty, x] \times (-\infty, y]$. Consider the union of the cells in $\mathcal{C}$. We will focus on constructing the *outer boundary* of the union that is shown in Figure 2. The outer boundary $\mathcal{B}$ is a 1-dimensional, monotone, and orthogonal curve. Specifically, $\mathcal{B}$ starts from $y = +\infty$ and consists of alternating vertical and horizontal segments. Let the sequence of endpoints of $\mathcal{B}$ be $c_1, d_1, c_2, d_2, \ldots, d_{t-1}$ and c_t. The c_i's are the *outer corners* and the d_i's are the *inner corners*. The number of points of P dominated by any outer corner (resp., inner corner) will be at most $2k$ (resp., equal to k). The k-level shallow cutting for 2-D dominance ranges can be constructed in $O(sort(N))$ I/Os. (Note that the $2k$ bound is tighter than the $10k$ bound we use for 3-D shallow cuttings.)

Offline-find-any procedure. The input to the *offline-find-any* problem is a collection of cells $\mathcal{C}$ in the k-level shallow cutting of P, for any $1 \leq k \leq n$, and a set Q of 3-D query points. In the offline-find-any procedure, for each $q \in Q$, the goal is to find *any* cell in $\mathcal{C}$ that contains q or report that none of them contain q. The distribution-sweeping approach in the survey paper of Vitter [21] can be used to solve this problem optimally.

▶ **Lemma 6.** *Given a set $\mathcal{C}$ of cells in 3-D and a set Q of query points in 3-D, the offline-find-any procedure requires $O\left(\left(\frac{|\mathcal{C}|+|Q|}{B}\right) \log_{M/B} \frac{|\mathcal{C}|}{B}\right)$ I/Os.*

3 Overview of previous construction algorithms

In this section we give an overview of a couple of previous constructions whose ideas will be required in our final solution.

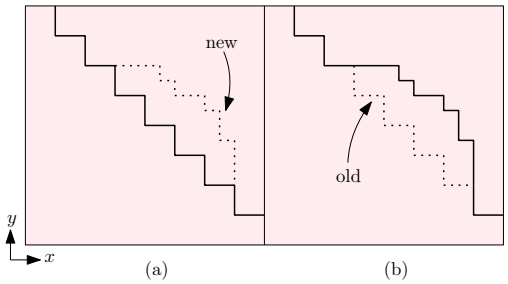


Figure 3 (a) Outer boundary (in bold) before the patching procedure. The dotted boundary is the new curve created during the patching procedure. (b) Outer boundary (in bold) after the patching procedure. The dotted curve was part of the outer boundary before patching. Each outer corner created during the patching procedure corresponds to the apex point of a 3-D cell in the k-level cutting of P.

3.1 Afshani and Tsakalidis's construction (AT-construction)

The problem becomes challenging in 3-D. At SODA'14, Afshani and Tsakalidis [5] obtained an optimal internal memory algorithm that constructs a k-level shallow cutting in $O(N \log_2 N)$ time. Their algorithm constructs the outer boundary $\mathcal{B}$ of the union of the cells in $\mathcal{C}$. In 3-D the outer boundary is a 2-dimensional, monotone, and orthogonal surface (see Figure 1(a)). The algorithm combines the sweep-plane approach with a method of maintaining a 2-D k-level shallow cutting under deletions. Let $z_1, \ldots, z_N$ be the sorted sequence of P in decreasing order of their z-coordinate values. Starting from $z = z_1$, a plane parallel to xy-plane is moved in the $-z$ direction. The *invariant* in the algorithm is to maintain a 2-D shallow cutting for the xy-projections of points of P below the sweep-plane, by ensuring that the depth of each (inner or outer) corner in the 2-D shallow cutting lies in the range $[k, 10k]$. When the sweep-plane reaches z_i, then z_i is deleted and if the invariant is violated, then the 2-D shallow cutting of xy-projections of $z_{i+1}, \ldots, z_n$ is updated via a procedure called *patching*. See Figure 3. The precise details of the patching procedure are not necessary for our final algorithm.

The running time of their algorithm can be summarized by the following equation:

$$T_1(N) = \underbrace{T_{sort}(N)}_{\text{sorting}} + \underbrace{\Theta(N) \cdot T_{pred}(N/k)}_{\text{bottleneck 1}} + \underbrace{O(N/k) \cdot T_{aux}(N)}_{\text{bottleneck 2}} \tag{1}$$

There are two bottlenecks in adapting their algorithm in the I/O-model. At any point in the sweep-plane, let $\mathcal{L}$ be the ordering of the corners in the 2-D shallow cutting based on their x-coordinate values. Each time a point $z_i \in P$ is deleted, a predecessor search is performed on $\mathcal{L}$. Also, the patching procedure leads to insertion and deletion of corners into $\mathcal{L}$. The predecessor search is performed to identify all the corners in $\mathcal{L}$ dominating z_i and then to update the conflict lists of those corners (at no extra asymptotic cost). This is the first bottleneck, since it is not possible to perform $\Theta(N)$ updates and $\Theta(N)$ predecessor queries to the list $\mathcal{L}$ in $sort(N)$ I/Os. (In fact, in [8] it was established that the easier problem of batched predecessor queries requires $T_{pred}(N) = \Omega(\log_B N)$ I/Os per query when the preprocessing cost is bounded by a polynomial in N.) It is not clear how to reduce the number of predecessor queries to $o(N)$, and the lower-bound discards the standard strategy in the I/O-model of performing predecessor search queries in batches.

The *second bottleneck* is the lack of I/O-efficient data structures for performing the patching procedure. The patching procedure involves constructing data structures for 3-D dominance x-selection and dominance y-selection queries on P, and querying them $O(N/k)$

times. For our purpose, the data structures have to be constructed in $O(sort(N))$ I/Os. In the full version, we present such data structures with query I/Os $T_{aux}(N) = O((N/B)^{\gamma})$ for some $\gamma \in (0,1)$ (Theorem 4). Therefore, the patching will require $O\left(\frac{N}{k}\left(\frac{N}{B}\right)^{\gamma}\right)$ I/Os, which is expensive.

3.2 Nekrich and Rahul's construction (NR-construction)

At SODA'23, Nekrich and Rahul [13] designed a data structure for 4-D dominance reporting query by constructing several shallow cuttings for 3-D dominance ranges. However, for their application, the conflict lists of the cells were not needed, and only the cells needed to be constructed. They modified the AT-construction to design an algorithm whose running time is proportional to the number of cells constructed (specifically $O(N \log_2 N)$ cells were constructed on different subsets of P in $O(N \log_2^5 N)$ time).

For a cell C in the shallow cutting with apex point (x, y, z), let $exp(C)$ be the largest z-coordinate such that $(x, y, exp(C))$ dominates less than k points of P. Then $exp(C)$ is defined as the *expiry* of C. For a cell C, its expiry $exp(C)$ can be computed via a 3-D dominance z-selection query on P. Similar to AT-construction approach, starting from $z = +\infty$, the algorithm sweeps a plane parallel to the xy-plane in the $-z$ direction and the same invariant is maintained. The key difference is in the *event points* visited by the sweep-plane algorithm. In the NR-construction, the event points will be the $exp(C)$ values of all cells C in the shallow cutting, which are stored in a simple priority-queue $\mathcal{Q}$.

At each time step, we find the event with the largest z-coordinate $z_{\max}$ in $\mathcal{Q}$ and set the z-coordinate of the sweep-plane to $z_{\max}$. When the cell C corresponding to z_{max} expires, (inner and outer) corners corresponding to C have to be removed from the 2-D shallow cutting at z_{max}. Therefore, the patching procedure is performed. After finishing the patching procedure, (a) the events corresponding to the new inner corners in the 2-D shallow cutting are inserted into $\mathcal{Q}$ and the events corresponding to the deleted corners in the 2-D shallow cutting are removed from $\mathcal{Q}$, (b) for each newly created outer corner (x, y) in the patching procedure, a new cell with apex point (x, y, z_{max}) gets *created*. We define $cre(C) = z_{max}$ for each cell C created. The algorithm finishes when the z-coordinate of the sweep-plane reaches $z = -\infty$. We omit the details of the NR-construction algorithm that are not needed to obtain our result. The following lemma summarizes the properties of the NR-construction that are relevant for our algorithm.

▶ **Lemma 7.** *Nekrich and Rahul's construction [13] of k-level shallow cuttings for 3-D dominance ranges can be summarized as follows:*

1. *It involves construction of a priority queue and 3-D dominance x, y, z-selection data structures.*[1]
2. *The number of updates and queries on the priority-queue is $O(N/k)$ and the number of 3-D dominance x, y, z-selection queries is $O(N/k)$.*
3. *In each selection query, if the query point is $q = (q_x, q_y, q_z)$, then the sweep-plane is at q_z and (q_x, q_y) is one of the corners in the 2-D shallow cutting of P at q_z after patching. The query integer $k' \leq 10k$ will be such that for an x-selection query, the reported point (q'_x, q_y, q_z) will have $q'_x > q_x$. However, for the y-selection and the z-selection queries, the reported points (q_x, q'_y, q_z) and (q_x, q_y, q'_z), respectively, will have $q'_y < q_y$ and $q'_z < q_z$.*

[1] Nekrich and Rahul [13] originally used *3-D dominance range counting* data structures as well, where the goal is to report $|P \cap q|$ for a 3-D dominance range q. However, both queries are almost equivalent: we can answer a 3-D dominance range selection query using $O(\log N)$ range counting queries (binary search). Conversely, we can also answer a 3-D dominance range counting query via $O(\log N)$ dominance z-selection queries.

The running time of the NR-construction can be summarized as follows:

$$T_2(N) = \underbrace{T_{\mathcal{Q}}\left(\frac{N}{k}\right)}_{\text{queue}} + \underbrace{O\left(\frac{N}{k}\right) \cdot T_{sel}(N)}_{\text{bottleneck}} \tag{2}$$

Similar to the second bottleneck in the AT-construction, the bottleneck in the NR-construction is the I/Os required to perform the dominance selection queries (with $O(sort(N))$ construction I/Os for the data structures, we have $O\left(\frac{N}{k}\right) \cdot T_{sel}(N) = O\left(\frac{N}{k}\left(\frac{N}{B}\right)^{\gamma}\right)$ I/Os which is expensive).

3.3 Other constructions

At ICALP'14, Afshani, Chan, and Tsakalidis [3] proposed an AT-construction variant (by sweeping upwards) for constructing shallow cuttings for 3-D dominance ranges in $O(N \log_2 \log_2 N)$ time in the word-RAM model. At STOC'96, Vengroff and Vitter [20] employed a structure that is similar to 3-D shallow cuttings to answer 3-D orthogonal range reporting queries. Unfortunately, the number of I/Os required to construct their data structure was not considered in their work.

Construction of shallow cuttings for halfspaces in 3-D has been another active area of research. However, typically the techniques and tools used for these problems (such as vertical decomposition, ε-cuttings, planar graph separators) have been different from the ones used for 3-D dominance ranges. For example, at SoCG'15, Chan and Tsakalidis [10] presented an optimal deterministic internal memory algorithm for the construction of shallow cuttings for halfspaces in 3-D. Their hierarchical construction relies on the fact that the cells of the shallow cutting do not intersect (to be precise, they are interior-disjoint). While this is true in the case of shallow cuttings for 3-D halfspaces, this property is not satisfied in our problem: cells of a k-level shallow cutting for 3-D dominance ranges are intersecting boxes. Also, in their hierarchy the levels go down by a constant factor which is not ideal for an I/O-efficient algorithm, whereas in our construction levels will go down at a faster rate.

4 Our algorithm for constructing k-level shallow cuttings

We are now ready to prove Theorem 1. Our algorithm follows the general approach used in [5, 13] and combines it with several other ideas. First, we construct a *hierarchy* of shallow cuttings such that the conflict list size of cells decreases exponentially. Second, somewhat counterintuitively, we use data structures with high query cost and fast pre-processing for 3-D dominance selection queries. The final algorithm is a combination of six pieces.

4.1 1st piece: Hierarchy of shallow cuttings

The main idea is to construct a hierarchy of shallow cuttings. We want to construct a k-level shallow cutting of P where $k \geq M/c$ and c is a sufficiently large constant. Let $t_i = (N/B)^{\delta^i}$ and $k_i = \max(Bt_i, k)$. We select the constant $\delta \in (0, 1)$ in such way that $k_i \leq k_{i-1}/10$ (for any i satisfying $k_i \geq M/c$). The algorithm works in stages (starting from stage 0). In the i-th stage, we construct the k_i-level shallow cutting. For the starting case of 0-th stage, since $k_0 = N$, the k_0-level shallow cutting consists of one cell containing all points of P. The algorithm terminates when $k_i = k$.

▶ **Lemma 8.** *The number of shallow cuttings constructed is* $r = O(\log_{M/B}(N/B))$.

Proof. Since $k_i = Bt_i \geq M/c$, we have $t_i \geq M/Bc$. Therefore, $t_{i+1} = t_i^{\delta} = t_i \cdot \frac{1}{t_i^{1-\delta}} \leq \frac{t_i}{(M/cB)^{1-\delta}}$. Unfolding the recursion, we obtain $t_{i+1} \leq \frac{t_0}{\left(\frac{M}{cB}\right)^{(i+1)(1-\delta)}}$. Setting $i \leftarrow \frac{c'}{1-\delta}\log_{M/cB}(N/B) - 1$, for $c' \geq 2$, we observe that $t_{i+1} \leq t_0/(N/B)^{c'} < 1$. This implies that $r = O(\frac{1}{1-\delta}\log_{M/B}(N/B)) = O(\log_{M/B}(N/B)$. ◀

4.2 2nd piece: Existence of parent cells

We will start with some definitions. For all $1 \leq i \leq r$, we will say that the k_{i-1}-level shallow cutting is the *parent* shallow cutting of the k_i-level shallow cutting. When the sweep-plane reaches z, then a cell C is *active* at z if $cre(C) \leq z \leq exp(C)$.

In the construction of the k_i-level shallow cutting, consider an outer corner (x_p, y_p) on the 2-D shallow cutting which got created when the sweep-plane is at position z. Then the *parent cell* of point $p = (x_p, y_p, z)$ is the cell C in the parent shallow cutting which satisfies the following two properties: (a) the apex point of C dominates p, and (b) the apex point of C has the smallest y-coordinate among all cells that are active at z.

▶ **Lemma 9.** *In the construction of the k_i-level shallow cutting (for $i \geq 1$), consider an outer corner (x_p, y_p) on the 2-D shallow cutting which got created when the sweep-plane is at position z. Then a parent cell always exists for the point (x_p, y_p, z).*

Proof. The depth of the point (x_p, y_p, z) will be less than or equal to $10k_i$ (an invariant maintained by the AT-construction), which is at most k_{i-1}. In the k_{i-1}-level shallow cutting construction, consider the 2-D shallow cutting when the sweep-plane reached z. This 2-D shallow cutting will have at least one outer corner, say p', which dominates (x_p, y_p) (by third property of k_{i-1}-level shallow cuttings). Let C be the 3-D cell in the k_{i-1}-level shallow cutting corresponding to the outer corner p'. Since C is active when the sweep-plane reached position z, it implies that the apex point of C dominates (x_p, y_p, z). This establishes property (a) of a parent cell, and hence, a parent cell exists. ◀

4.3 3rd piece: Query-expensive structures on small-sized inputs

A data structure built on N' points is query-expensive if the the number of I/Os required to answer a query is of the form $O((N'/B)^{\gamma})$ or $O((N'/B)^{\gamma} + t/B)$. Our next idea is to construct such query-expensive data structures, but only on small-sized inputs. Specifically, for the k_i-level shallow cutting, we will ensure that the a 3-D dominance x, y, z-selection query requires only $O(k_i/B)$ I/Os, and since the number of such selection queries performed is bounded by $O(N/k_i)$ (point 2 in Lemma 7), overall the selection queries will require only $O(N/B)$ I/Os. Similarly, we will ensure that generating the conflict list of a cell requires only $O(k_i/B)$ I/Os, and since there are $O(N/k_i)$ cells in the k_i-level cutting, generating all the conflict lists will require only $O(N/B)$ I/Os. The existence of parent cells and the following lemma will let us achieve the goal of utilizing query-expensive data structures I/O-efficiently.

▶ **Lemma 10.** *In the construction of the k_i-level shallow cutting (for $i \geq 1$), consider the sweep-plane at any time z. If patching is performed at time z, then let $p = (x_p, y_p)$ be a new outer corner created with parent cell C. Then the 3-D dominance x, y, z-selection queries with query point $p = (x_p, y_p, z)$ will output a point that lies inside C.*

Proof. This is trivial to see for a y-selection query since p already lies inside C and hence, the point reported by the query will also lie inside C (see Lemma 7). Similarly, it holds for a z-selection query as well. The non-trivial case is the x-selection query.

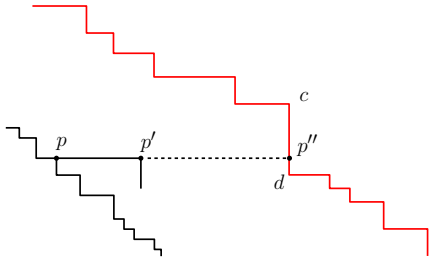


Figure 4 An example of x-selection query in Lemma 10. The point p is an outer corner of a 2-D shallow cutting of the k_i-level. The corner c is the outer corner of the parent staircase corresponding to the parent cell C of p, since c has the smallest y-coordinate among all apex points of the parent staircase dominating p.

Let p'' denote the point where the horizontal ray from p hits the 2-D shallow cutting of the parent at time z. See Figure 4. Let d be the inner corner immediately below p'' and c be the outer corner immediately above p''. Since the depth of d is at least k_{i-1}, the depth of p'' will also be at least k_{i-1}. Consider the 3-D dominance x-selection query with query point p and parameter $k' \leq 10k_i \leq k_{i-1}$. This implies that the output, p', will have an x-coordinate less than or equal to p''. Now the cell corresponding to the outer corner c is the parent cell of p and contains p' as well. ◀

▶ **Lemma 11.** *Assume that the k_{i-1}-level shallow cutting of P has been constructed in the $(i-1)$-th stage. Ignoring the I/Os required to find the parent cells, in the i-th stage, the k_i-level shallow cutting can be constructed in $O\left(\frac{N}{k_i}\log_B\frac{N}{k_i} + \frac{N}{B}\log_{M/B}\frac{k_{i-1}}{B}\right)$ I/Os.*

Proof. For each cell in the k_{i-1}-level shallow cutting, based on its conflict list we construct data structures supporting 3-D dominance reporting and 3-D dominance x, y, z-selection queries (Theorem 4) that requires $O((N/k_{i-1}) \cdot \frac{k_{i-1}}{B}\log_{M/B}(\frac{k_{i-1}}{B})) = O\left(\frac{N}{B}\log_{M/B}\frac{k_{i-1}}{B}\right)$ I/Os. By point 2 in Lemma 7, $O(N/k_i)$ x, y, z-selection queries will be performed. For each selection query, if the cost of finding the parent cell of the query point is ignored, then by Lemma 10 the answer to the query can be obtained by querying the x, y, z-selection data structure built at the parent cell. The query I/Os of each selection query will then be $O((k_{i-1}/B)^\gamma)$. Therefore, the number of I/Os performed to answer the selection queries is $O\left(\frac{N}{k_i}\left(\frac{k_{i-1}}{B}\right)^\gamma\right) = O\left(\frac{N}{Bt_i}\cdot t_{i-1}^\gamma\right) = O\left(\frac{N}{Bt_i}\cdot t_{i-1}^\delta\right) = O\left(\frac{N}{B}\right)$, by choosing $\gamma < \delta$.

By point 1 in Lemma 7, the number of operations on the priority-queue are $O(N/k_i)$ and using a vanilla B-tree as a priority-queue $O\left(\frac{N}{k_i}\log_B\frac{N}{k_i}\right)$ I/Os are performed.

Finally, for each cell C in the k_i-level cutting, its conflict list is generated by querying its parent cell's 3-D dominance reporting data structure. The number of I/Os performed is $O\left(\left(\frac{k_{i-1}}{B}\right)^\gamma + \frac{k_i}{B}\right) = O(k_i/B)$. Since the number of cells in the k_i-level cutting is $O(N/k_i)$, the total number of I/Os required to generate the conflict lists is $O(N/B)$. ◀

4.4 4th piece: Finding parent cells efficiently

By point 2 in Lemma 7, $O(N/k_i)$ x, y, z-selection queries are performed in the construction of a k_i-level shallow cutting. To perform each selection query, we first need an efficient routine to find the parent cell. Once the parent cell is found, then the selection query is performed on the data structure corresponding to the parent cell.

We will use partially persistent B-trees [7] to store the cells in the k_{i-1}-level shallow cutting. Versions of the partially persistent data structure are parameterized by z-coordinates: a cell is inserted (resp. deleted) at time t if the corresponding outer corner is added to the 2-D shallow cutting (resp. removed from the 2-D shallow cutting) when the z-coordinate of the sweep-plane is equal to t. This data structure supports successor queries with respect to y- coordinates of its apex points: for any pair (z_q, y_q) we can find the cell with the smallest y-coordinate among all cells of the parent shallow cutting that are active at time z_q and whose y-coordinate is not smaller than y_q.

▶ **Lemma 12.** *In the construction of k_i-level shallow cutting, the number of I/Os performed to compute the parent cells is $\left(\frac{N}{k_i}\log_B \frac{N}{k_{i-1}}\right)$.*

Proof. The number of cells in the k_{i-1}-level shallow cutting is $O(N/k_{i-1})$ and using persistent B-trees, a successor query and an update query can be supported in $O(\log_B(N/k_{i-1}))$ I/Os. By point 2 in Lemma 7, the total number of x, y, z-selection queries will be $O(N/k_i)$, and hence, the number of I/Os performed to compute the parent cells is $\left(\frac{N}{k_i}\log_B \frac{N}{k_{i-1}}\right)$. ◀

4.5 5th piece: Obtaining logarithmic bounds with the base M/B

To obtain $O(sort(N))$ I/Os bound, we need the base in the logarithm to be M/B. To circumvent the fact that finding parent cells and operations on the priority queue require $O(\log_B(N/k))$ query I/Os, we exploit the fact that k is large.

▶ **Lemma 13.** *For $k = \Omega(M)$, we have $O\left(\frac{N}{k}\log_B \frac{N}{k}\right) = O\left(\frac{N}{B}\log_{M/B} \frac{N}{B}\right)$.*

Proof. First, we will establish that $B\log(M/B) \le cM\log B$, where c is a sufficiently large constant. When $M \le B^c$, we have $cM\log B = M\log B^c \ge M\log M \ge B\log(M/B)$. When $M > B^c$, we have $cM\log B \ge cB\cdot M^{1-1/c}\log B \ge cB\cdot M^{1-1/c} \ge cB\log M \ge cB\log(M/B)$. Using this fact, we have

$$\frac{N}{k}\cdot\frac{\log(N/k)}{\log B} \le \frac{N}{M}\cdot\frac{\log(N/B)}{\log B} \le O\left(\frac{N}{B}\frac{\log(N/B)}{\log(M/B)}\right). \quad ◀$$

In summary, the running time of our algorithm is

$$T_3(N) = O\left(T_{\mathcal{Q}}\left(\frac{N}{k}\right) + \sum_{i=1}^{r}\frac{N}{k_i}\cdot T_{sel}(k_{i-1}) + \frac{N}{k}\cdot\log_B\frac{N}{k}\right) \tag{3}$$

▶ **Lemma 14.** *For $k = \Omega(M)$, the k-level shallow cutting for 3-D dominance ranges on N points can be constructed in $O\left(\frac{N}{B}\log_{M/B}\left(\frac{N}{B}\right)\right)$ I/Os.*

Proof. By combining Lemma 11 and Lemma 12, the number of I/Os required to construct a k_i-level shallow cutting is $O\left(\frac{N}{k_i}\log_B\frac{N}{k_i} + \frac{N}{B}\log_{M/B}\frac{k_{i-1}}{B}\right)$. Therefore,

$$\sum_{i=1}^{r} O\left(\frac{N}{k_i}\log_B\frac{N}{k_i} + \frac{N}{B}\log_{M/B}\frac{k_{i-1}}{B}\right) = O\left(\frac{N}{k}\log_B\frac{N}{k} + \frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

By Lemma 13, this can be bounded by $O\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$. ◀

4.6 Final piece: Handling Small Values of k

We will need a different procedure to handle $k = O(M)$. We will perform a two-level cutting. First, we will construct an (M/c)-level shallow cutting of P using Lemma 14. For each cell, say C, in the (M/c)-level, load its *entire* conflict list into the internal memory. This can be done I/O-efficiently since the size of the conflict list is $O(M)$. Next run an internal memory algorithm to compute the k-level shallow cutting of the conflict list of C. The final output is the union of the shallow cuttings obtained from each cell in the (M/c)-level.

▶ **Lemma 15.** *Suppose that we are given an (M/c)-level shallow cutting of P for some constant $c > 1$. Then for $k < M/c$, the k-level shallow cutting can be constructed in $O(N/B)$ I/Os.*

Proof. We will establish that the three properties of a k-level shallow cutting of P are satisfied by the algorithm. The number of cells constructed in the final cutting is $O(N/M) \cdot O(M/k) = O(N/k)$ (first property). It is easy to verify that the number of points of P lying inside each cell in the final cutting is $O(k)$ (second property). Consider a point q that dominates less than or equal to k points in P. By the third property of M/c-level shallow cutting of P, there exists a cell C that contains q. The number of points in the conflict list of C dominated by q will still be less than or equal to k. Therefore, in the k-level shallow cutting of the conflict list of C, there will exist some cell that contains q (third property).

Loading the conflict lists of all the cells in the (M/c)-level shallow cutting requires $O(N/M) \cdot O(M/B) = O(N/B)$ I/Os. For each cell in the (M/c)-level shallow cutting, writing the k-level shallow cutting of its conflict-list into external memory requires $O(M/B)$ I/Os. Thus, over all the cells in the (M/c)-level shallow cutting, it will require $O(N/M) \cdot O(M/B) = O(N/B)$ I/Os. ◀

Combining Lemma 14 and Lemma 15 completes the proof of Theorem 1.

5 Application-1: Offline 3-D dominance approximate counting

Recall that P is a set of N points in 3-D and Q is a set of query 3-D dominance ranges. Consider a parameter ε with a fixed value in $(0, 1)$. In the offline 3-D dominance approximate counting problem, the goal is to report a value in the range $[(1 - \varepsilon)|P \cap q|, |P \cap q|]$, for all $q \in Q$. We will prove Theorem 2 in this section.

5.1 Data structure and query algorithm

The overall data structure is recursive and we start at the root with pointset P. Construct a $(N^{2/3}B^{1/3})$-level cutting $\mathcal{C}$ based on P (Theorem 1). Also, construct an additive error counting structure based on P (Theorem 5). Finally, for each cell $C \in \mathcal{C}$, recurse on P_C, where P_C is the conflict list of C. The recursion stops when the size of pointset becomes less than or equal to M/c, where c is a sufficiently large constant.

The query algorithm will start from the root of the data structure. A point in Q is *deep* if no cell in $\mathcal{C}$ contains it; otherwise, it is *shallow*. Using the offline-find-any procedure on $\mathcal{C}$ and Q, we will classify each point in Q as shallow or deep. The result for the deep query points are obtained by querying the additive error counting structure. If a point $q \in Q$ is shallow, then it is *assigned* to one of the cells containing it. The shallow points are handled recursively. Specifically, for each cell $C \in \mathcal{C}$, *recurse* on Q_c, where $Q_c \subseteq Q$ is the set of shallow query points assigned to C.

To handle the base case, where $|P| \leq M/c$, first the points in P are loaded into the memory. Then the query objects are streamed into the main memory one block at a time, and the results for these queries are obtained without any additional I/Os and written to the external memory.

5.2 Analysis

We will first bound the number of I/Os performed by the algorithm.

▶ **Lemma 16.** *The number of I/Os required to construct the data structure is* $O\left(\frac{|P|}{B}\left(\log_{M/B}\frac{|P|}{B}\right)^{O(1)}\right)$.

Proof. The $(N^{2/3}B^{1/3})$-level cutting (Theorem 1) and the additive error counting structure (Theorem 5) can be constructed in $O(sort(|P|)$ I/Os. Therefore, the total number of I/Os required to construct the data structure is:

$$T(N) = \begin{cases} c_1\left(\frac{N}{B}\right)^{1/3} \cdot T(N^{2/3}B^{1/3}) + \frac{c_2 N}{B}\log_{M/B}\frac{N}{B}, \text{if } N \geq M/c \\ O(N/B), \text{ otherwise} \end{cases}$$

where $c_1(N/B)^{1/3}$ is the number of cells in the cutting and c_2 is the constant inside $O(sort(N))$. We will establish that $T(N) \leq \frac{c_3 N}{B}\left(\log_{M/B}\frac{N}{B}\right)^{c_4}$, where $c_3 = 2c_2$ and $c_1\left(\frac{2}{3}\right)^{c_4} = \frac{1}{2}$. As such,

$$\begin{aligned} T(N) &\leq c_1\left(\frac{N}{B}\right)^{1/3} \cdot c_3\frac{N^{2/3}B^{1/3}}{B}\left(\log_{M/B}\frac{N^{2/3}B^{1/3}}{B}\right)^{c_4} + \frac{c_2 N}{B}\log_{M/B}\frac{N}{B} \\ &\leq c_1 c_3\frac{N}{B}\cdot\left(\frac{2}{3}\right)^{c_4}\left(\log_{M/B}\frac{N}{B}\right)^{c_4} + \frac{c_2 N}{B}\log_{M/B}\frac{N}{B} \\ &\leq \frac{c_2 N}{B}\left(\log_{M/B}\frac{N}{B}\right)^{c_4} + \frac{c_2 N}{B}\left(\log_{M/B}\frac{N}{B}\right)^{c_4} = \frac{c_3 N}{B}\left(\log_{M/B}\frac{N}{B}\right)^{c_4} \end{aligned}$$

◀

▶ **Lemma 17.** *The number of I/Os performed by the query algorithm is* $O_\varepsilon\left(\frac{|P|}{B}\left(\log_{M/B}\frac{|P|}{B}\right)^{O(1)} + \frac{|Q|}{B}\log_{M/B}\frac{N}{B}\right)$.

Proof. By Theorem 5 and Lemma 6, the number of I/Os performed by the query algorithm at the root is $O_\varepsilon\left(sort(N) + sort(|Q|) + \left(\frac{|\mathcal{C}|+|Q|}{B}\right)\log_{M/B}\frac{|\mathcal{C}|}{B}\right)$, which can be bounded by $O_\varepsilon\left(sort(N) + \frac{|Q|}{B}\log_{M/B}\frac{N}{B}\right)$. The $O_\varepsilon(sort(N))$ I/Os can be charged to the construction of the data structure, and hence, by the proof of Lemma 16, it will be bounded by $O_\varepsilon\left(\frac{|P|}{B}\left(\log_{M/B}\frac{|P|}{B}\right)^{O(1)}\right)$ in the overall query algorithm.

Therefore, we will focus on the $O_\varepsilon\left(\frac{|Q|}{B}\log_{M/B}\frac{N}{B}\right)$ term. As such, the amortized number of I/Os performed for a *single* query point is $O_\varepsilon\left(\frac{1}{B}\log_{M/B}\frac{N}{B}\right)$. Including recursion, the number of I/Os performed to answer a single query is:

$$q(N) \leq q(N^{2/3}B^{1/3}) + O_\varepsilon\left(\frac{1}{B}\log_{M/B}\frac{N}{B}\right) = O_\varepsilon\left(\frac{1}{B}\log_{M/B}\frac{N}{B}\right),$$

which holds because we end up with a geometrically decreasing series. ◀

▶ **Lemma 18.** *For each $q \in Q$, the algorithm returns a $(1-\varepsilon)$-approximation of $|P \cap q|$.*

Proof. For any deep query point q at the root, the value n_q reported by the additive error counting structure will be such that $n_q \geq |P \cap q| - \varepsilon N^{2/3}B^{1/3} \geq (1-\varepsilon)|P \cap q|$, since $|P \cap q| \geq N^{2/3}B^{1/3}$ for a deep point (third property of shallow cuttings). The same argument applies at each node in the data structure (w.r.t. deep queries at that node). ◀

References

1 Peyman Afshani. On dominance reporting in 3D. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 41–51, 2008. doi:10.1007/978-3-540-87744-8_4.

2 Peyman Afshani, Lars Arge, and Kasper Dalgaard Larsen. Orthogonal range reporting in three and higher dimensions. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 149–158, 2009. doi:10.1109/FOCS.2009.58.

3 Peyman Afshani, Timothy M. Chan, and Konstantinos Tsakalidis. Deterministic rectangle enclosure and offline dominance reporting on the RAM. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 77–88, 2014. doi:10.1007/978-3-662-43948-7_7.

4 Peyman Afshani, Chris H. Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Computational Geometry*, 43(8):700–712, 2010. doi:10.1016/J.COMGEO.2010.04.003.

5 Peyman Afshani and Konstantinos Tsakalidis. Optimal deterministic shallow cuttings for 3d dominance ranges. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1389–1398, 2014. doi:10.1137/1.9781611973402.102.

6 Pankaj K. Agarwal, Lars Arge, Jeff Erickson, Paolo Giulio Franciosa, and Jeffrey Scott Vitter. Efficient searching with linear constraints. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 169–178, 1998. doi:10.1145/275487.275506.

7 Bruno Becker, Stephan Gschwind, Thomas Ohler, Bernhard Seeger, and Peter Widmayer. An asymptotically optimal multiversion B-tree. *The VLDB Journal*, 5(4):264–275, 1996. doi:10.1007/S007780050028.

8 Michael A. Bender, Martin Farach-Colton, Mayank Goswami, Dzejla Medjedovic, Pablo Montes, and Meng-Tsung Tsai. The batched predecessor problem in external memory. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 112–124, 2014. doi:10.1007/978-3-662-44777-2_10.

9 Timothy M. Chan, Yakov Nekrich, Saladi Rahul, and Konstantinos Tsakalidis. Orthogonal point location and rectangle stabbing queries in 3-d. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 31:1–31:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.31.

10 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. In *International Symposium on Computational Geometry (SoCG)*, pages 719–732, 2015. doi:10.4230/LIPIcs.SOCG.2015.719.

11 Roberto Grossi and Giuseppe F. Italiano. Efficient cross-trees for external memory. In *External Memory Algorithms*, volume 50, pages 87–106. DIMACS/AMS, 1998. doi:10.1090/DIMACS/050/04.

12 Kothuri Venkata Ravi Kanth and Ambuj K. Singh. Optimal dynamic range searching in non-replicating index structures. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 257–276, 1999. doi:10.1007/3-540-49257-7_17.

13 Yakov Nekrich and Saladi Rahul. 4d range reporting in the pointer machine model in almost-optimal time. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1862–1876, 2023. doi:10.1137/1.9781611977554.CH71.

14 Mihai Patrascu. Succincter. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 305–313, 2008. doi:10.1109/FOCS.2008.83.

15 Saladi Rahul. Improved bounds for orthogonal point enclosure query and point location in orthogonal subdivisions in $\mathbb{R}^3$. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 200–211, 2015. doi:10.1137/1.9781611973730.15.

16 Saladi Rahul and Yufei Tao. On top-k range reporting in 2d space. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 265–275, 2015. doi:10.1145/2745754.2745777.

17 Saladi Rahul and Yufei Tao. Efficient top-k indexing via general reductions. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 277–288, 2016. doi:10.1145/2902251.2902290.

18 John T. Robinson. The K-D-B-tree: A search structure for large multidimensional dynamic indexes. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 10–18, 1981. doi:10.1145/582318.582321.

19 Yufei Tao. Indexability of 2d range search revisited: constant redundancy and weak indivisibility. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 131–142, 2012. doi:10.1145/2213556.2213577.

20 Darren Erik Vengroff and Jeffrey Scott Vitter. Efficient 3-d range searching in external memory. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 192–201, 1996. doi:10.1145/237814.237864.

21 Jeffrey Scott Vitter. Algorithms and data structures for external memory. *Foundation and Trends in Theoretical Computer Science*, 2(4):305–474, 2006. doi:10.1561/0400000014.

A Persistent Version of Latschev's Theorem

Steve Oudot
Inria Saclay, and École Polytechnique, Palaiseau, France

Lukas Waas
University of Oxford, UK

Abstract

Latschev's theorem provides sufficient conditions on a metric space M and $\delta > 0$ for the homotopy type of M to agree with that of the Vietoris-Rips complex $\mathcal{R}^\delta(\mathbb{M})$ of any nearby space $\mathbb{M}$ in the Gromov-Hausdorff distance. We prove a persistent version of this theorem, providing sufficient conditions on a pair $(M, f\colon M \to \mathbb{R}^N)$ and $\delta > 0$ for the persistent homotopy type of the sublevel set filtration of (M, f) to be interleaved with that of the function-Rips complex $\mathcal{R}^\delta(\mathbb{M}^\bullet)$ of any nearby pair $(\mathbb{M}, \mathbb{f})$. In particular, our result answers a longstanding question on the related topic of estimating sublevel set persistent homology from finite point samples.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases Topological data analysis (TDA), metric geometry, Vietoris-Rips complex, homotopy theory, multi-parameter persistent homology

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.82

Related Version *Full Version: "Function-Rips complexes in persistent homotopy theory: Stability and Latschev theorems"*: `https://doi.org/10.48550/arXiv.2603.23460` [31]

Funding *Lukas Waas*: Lukas Waas acknowledges support from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2181/1 – 390900948 (the Heidelberg STRUCTURES Excellence Cluster). Furthermore, Lukas Waas is a member of the Oxford/Max Planck collaboration and this research was funded in in part by EPSRC international centre to centre collaboration grant EP/Z531224/1.

Acknowledgements This work started when the two authors attended the *TDA week* at Kyoto University in June 2025. It then continued as the two authors participated in the thematic program *Topological Data Analysis, Persistence And Representation Theory Intertwined* (TP25TD) at the Okinawa Institute of Science and Technology, from June to August 2025.

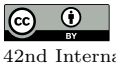
1 Introduction

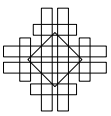
This work connects three lines of research in TDA and metric topology. The first – and main – one aims at generalizing Latschev's result about the homotopy type of Vietoris-Rips complexes built on Gromov-Hausdorff approximations of compact Riemannian manifolds.

▶ **Theorem** (Latschev [25]). *Let M be a closed Riemannian manifold. Then there exists $\delta_0 > 0$ such that, for every $0 < \delta \leq \delta_0$, there exists an $\varepsilon > 0$ such that, for any metric space $\mathbb{M}$ with Gromov-Hausdorff distance to M less than ε, M is homotopy equivalent to the geometric realization $|\mathcal{R}^\delta(\mathbb{M})|$ of the Vietoris-Rips complex $\mathcal{R}^\delta(\mathbb{M})$ of parameter δ.*

To our knowledge, this result has been extended in the following directions:

- Replacing Riemannian manifolds with spaces of curvature bounded above [23];
- Quantifying the upper bounds on ε and δ [27, 29];
- Letting δ go past its upper bound on specific spaces, for instance the circle [1] or ellipses [3];
- Considering variants of the Vietoris-Rips complex, such as the selective Rips complex [26].


© Steve Oudot and Lukas Waas;
licensed under Creative Commons License CC-BY 4.0



Here, we extend Latschev's result in another direction: To spaces equipped with filtration functions (Theorem 1 below). In this setting, the role of the homotopy type of M is taken by the so-called *persistent homotopy type* of the sublevel set filtration of a function $f\colon M \to \mathbb{R}^N$, denoted $M^\bullet$ (Notation 19). The role of the Vietoris-Rips complex of $\mathbb{M}$ is now taken by the filtered simplicial complex $\mathcal{R}^\delta(\mathbb{M}^\bullet)$, called the *function-Rips* complex. It is obtained by filtering the ordinary Rips complex of fixed parameter δ, $\mathcal{R}^\delta(\mathbb{M})$, through the Rips-complexes (at scale δ) of the sublevel sets of a function $\mathbb{f}\colon \mathbb{M} \to \mathbb{R}^N$ (Example 12). Finally, the role of correspondences (for the Gromov-Hausdorff distance) is taken by so-called *filtered correspondences* (Definition 32), and the role of homotopy equivalences is taken by a notion of *interleavings in the persistent homotopy category* (Definition 28, see also [24]). In the following, M denotes a compact metric space with curvature bounded above by some $\kappa \in \mathbb{R}$ (Recollection 5) and $f\colon M \to \mathbb{R}^N$ a componentwise 1-Lipschitz map. Finally, ρ_M^κ denotes the minimum of the convexity radius ϱ_M of M (see Recollection 4) and of either $\frac{\pi}{4\sqrt{\kappa}}$ if $\kappa > 0$ or ∞ if $\kappa \leq 0$.

▶ **Theorem 1** (Persistent Latschev's theorem). *Let $0 < \delta \leq \rho_M^\kappa$ and $\varepsilon \leq \min\{\rho_M^\kappa - \delta, \frac{1}{7}\delta\}$. Then, for any filtered ε-correspondence $(M, f) \approx_\varepsilon (\mathbb{M}, \mathbb{f})$, there is a $\frac{7}{4}\delta$-interleaving $M^\bullet \simeq_{\frac{7}{4}\delta} \mathcal{R}^\delta(\mathbb{M}^\bullet)$ in the persistent homotopy category.*

Note that, in contrast to the setup of Latschev's theorem, we cannot hope to get a homotopy equivalence between filtered spaces here. This is because the individual sublevel sets of f may be highly pathological even though M itself satisfies strict regularity conditions. Note also that we cannot expect interleavings at the level of topological spaces: any such interleaving would induce homeomorphisms at infinity, which generally do not exist.

In the special case $N = 0$ (i.e., $\mathbb{R}^N$ is a singleton), the homotopy theoretic interleaving $M^\bullet \simeq_{\frac{7}{4}\delta} \mathcal{R}^\delta(\mathbb{M}^\bullet)$ is just an isomorphism in the homotopy category, i.e., a zig-zag of weak homotopy equivalences. So, by an application of Whitehead's theorem, Theorem 1 recovers the quantitative version of Latschev's theorem given in [29], with the same bounds on δ and ε. As Theorem 1 is phrased at the persistent homotopy level, it also induces an interleaving at the persistent homology level. This connects our result to another line of research, the aim of which is to estimate the persistent homology of functions from finite point samples. In [14], the authors proposed an estimator for the persistent homology of $L^\bullet$, for a Lipschitz map $g\colon L \to \mathbb{R}$, from a finite sampling $\mathbb{L} \subset L$ of its domain L. The proof technique leveraged in this line of work made use of the interleaving between Vietoris-Rips and Čech complexes, and ultimately led to an estimator given by the image of morphisms of persistence modules $\mathrm{im}(H_*\mathcal{R}^\delta(\mathbb{L}^\bullet) \to H_*\mathcal{R}^{2\delta}(\mathbb{L}^\bullet))$, where $\mathbb{L}$ is filtered by $g|_{\mathbb{L}}$. This estimator was then specialized to the case where g is a density estimator in the context of unsupervised learning, yielding the clustering algorithm ToMATo [15]. It was later extended to more general noise models [10] and, more recently, to $\mathbb{R}^N$-valued maps:

▶ **Theorem** ([4]). *Let (L, g) be such that L is a compact metric space with $\varrho_L > 0$ and g is 1-Lipschitz. Let $\delta < \frac{1}{2}\varrho_L$ and $\varepsilon \leq \frac{1}{2}\delta$. Then, for any subset $\mathbb{L} \subset L$ that is ε-dense in L, there is a 2δ-interleaving $H_*(L^\bullet) \simeq_{2\delta} \mathrm{im}(H_*\mathcal{R}^\delta(\mathbb{L}^\bullet) \to H_*\mathcal{R}^{2\delta}(\mathbb{L}^\bullet))$ of persistence modules.*

Theorem 1 complements this result by showing that, under a more restrictive regularity condition on M (namely: that M has curvature bounded above), a single filtered Rips complex is enough to estimate the persistent homology of f, via a different proof approach that does not proceed through the interleaving between Vietoris-Rips and Čech complexes. The question of whether this is possible had been open since the beginning of this line of work, and it has important implications, including algorithmic ones. For instance, computing

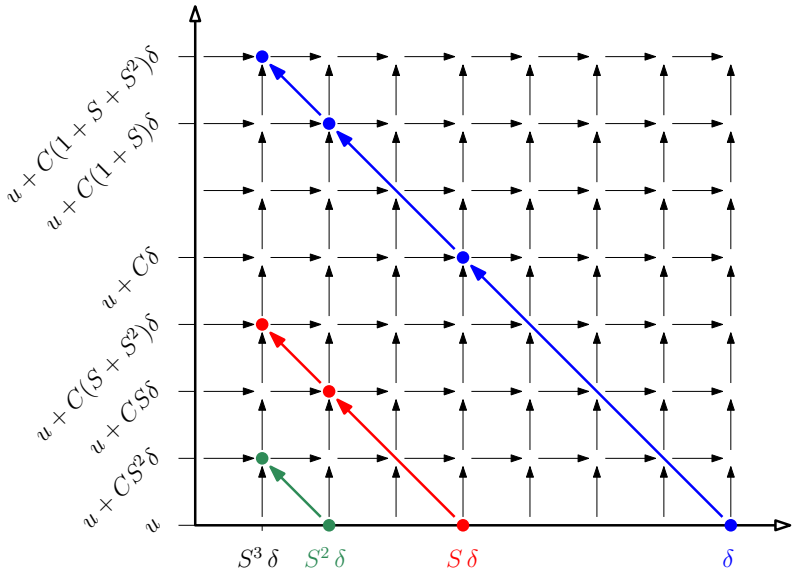


Figure 1 Shrinking transformation arrows (colored) and the object's structure morphisms (black).

a free presentation of $\mathrm{im}(H_*\mathcal{R}^\delta(\mathbb{M}^\bullet) \to H_*\mathcal{R}^{2\delta}(\mathbb{M}^\bullet))$ involves computing the free cover of a certain pullback for which efficient specialized algorithms exist only when $N = 1$ or 2 (see [4]); otherwise, one must resort to Schreyer's algorithm [33] with doubly exponential complexity in N. By contrast, free presentations of $H_*\mathcal{R}^\delta(\mathbb{M}^\bullet)$ can be computed efficiently for any $N \geq 1$ via specialized algorithms for the computation of Gröbner bases [18].

Similarly to the proof of the classical Latschev theorem, the proof of our persistent version (Theorem 1) is a two-step process. Firstly, one needs a persistent version of Hausmann's theorem allowing us to relate $M^\bullet$ to $\mathcal{R}^\delta(M^\bullet)$.

▶ **Theorem 2** (Persistent Hausmann's Theorem). *Let $0 < \delta \leq \rho_M^\kappa$. Then there is a δ-interleaving $M^\bullet \simeq_{\frac{3}{4}\delta} \mathcal{R}^\delta(M^\bullet)$ in the persistent homotopy category.*

This result, proven in the full version [31], is a rather straightforward consequence of a powerful new technique called metric thickening, which provides a more geometrical model for the Vietoris-Rips complex [2, 19]. The crucial and technically more difficult part of proving Theorem 1 is to relate $\mathcal{R}^\delta(M^\bullet)$ and $\mathcal{R}^\delta(\mathbb{M}^\bullet)$. Here, we prove the following result:

▶ **Theorem 3** (Approximation of function-Rips persistent homotopy type). *Let $0 < \delta \leq \rho_M^\kappa$ and $\varepsilon \leq \min\{\rho_M^\kappa - \delta, \frac{1}{7}\delta\}$. Then, for any filtered ε-correspondence $(\mathbb{M}, \mathbb{f}) \approx_\varepsilon (M, f)$, there is a δ-interleaving $\mathcal{R}^\delta(\mathbb{M}^\bullet) \simeq_\delta \mathcal{R}^\delta(M^\bullet)$ in the persistent homotopy category.*

To derive this result, we first prove a bivariate interleaving $\mathcal{R}^\bullet(M^\bullet) \simeq_\varepsilon \mathcal{R}^\bullet(\mathbb{M}^\bullet)$ from the filtered correspondence $(\mathbb{M}, \mathbb{f}) \approx_\varepsilon (M, f)$ in Section 3 (Proposition 34). Then, to leverage this interleaving, in Section 4, we introduce and study the general notion of a *shrinking transformation*: a transformation of the form $F^{\bullet_1,\bullet_2} \to F^{S\bullet_1,\bullet_2+C\bullet_1}$, with $C \geq 0$ and $0 \leq S < 1$, allowing one to move backwards in one parameter at the cost of increasing the other in a bivariate persistent object $F^{\bullet,\bullet}$, as illustrated in Figure 1. Shrinking transformations can be used to turn interleavings in two variables $F^{\bullet,\bullet} \simeq_\varepsilon G^{\bullet,\bullet}$ into interleavings of the form $F^{\delta,\bullet} \simeq_{C\delta+(1+C)\varepsilon} G^{\delta,\bullet}$ (Theorem 44). Then, in Section 5 we work out conditions on M that guarantee the existence of such a shrinking transformation on $\mathcal{R}^\bullet(M^\bullet)$ (Theorem 51). One incarnation of these conditions is verified in the quantitative version of Latschev's theorem [29]

and produces a shrinking transformation with $C = S = \frac{3}{4}$. From these conditions, combined with Proposition 34 and Theorem 44, we get Theorem 3. Together with Theorem 2, this gives the $\frac{7}{4}\delta$-interleaving $M^\bullet \simeq_{\frac{7}{4}\delta} \mathcal{R}^\delta(\mathbb{M}^\bullet)$ of Theorem 1.
We note that most of our techniques, particularly Theorems 44 and 51, are quite general and seem widely applicable. Indeed, in the full version of this article [31], we will use them to prove local stability theorems for the persistent homotopy types of function-Rips complexes.

2 Background

2.1 Spaces of curvature bounded above

▶ **Recollection 4** (Geodesics and convexity)**.** Let M be a metric space with metric d. By a (constant speed) *geodesic*, we mean a map $\gamma\colon [0,1] \to M$ such that $d(\gamma(s),\gamma(t)) = |s-t|\, d(\gamma(0),\gamma(1))$ for all $s \le t \in [0,1]$. We say that a subset $A \subset M$ is *convex*, if every pair of points $x, y \in A$ is connected by a unique constant speed geodesic γ from x to y in M, and furthermore every such geodesic is contained in A. We denote

$$\varrho_M(x) := \sup\{r > 0 \mid B_s(x) \text{ is convex, for all } s < r\}$$

the convexity radius at $x \in M$. The *convexity radius of* M is defined as $\varrho_M := \inf_{x\in M} \varrho_M(x)$.

▶ **Recollection 5** (Curvature bounded above)**.** Given $\kappa \in \mathbb{R}$, we call M_κ the following space:
1. for $\kappa > 0$, the sphere of radius $\frac{1}{\sqrt{\kappa}}$ in $\mathbb{R}^3$, equipped with the geodesic distance;
2. for $\kappa = 0$, the euclidean plane $\mathbb{R}^2$;
3. for $\kappa < 0$, the rescaling of the hyperbolic plane $\mathbb{H}$ by $\frac{1}{\sqrt{-\kappa}}$.

We denote by ϖ_κ the diameter of M_κ ($\varpi_\kappa = \frac{\pi}{\sqrt{\kappa}}$ if $\kappa > 0$ and $\varpi_\kappa = \infty$ if $\kappa \le 0$). We say that M *is a* CAT(κ) *space* if any two points with distance at most ϖ_κ are connected by a geodesic, and furthermore, every geodesic triangle of perimeter smaller than $2\varpi_\kappa$ is *no thicker than a comparison triangle in* M_κ (see Figure 2 and [31, 9] for a rigorous definition). A metric space M is called *of curvature bounded above by* κ - or a CBA(κ) space for short - if every point $x \in M$ admits a neighborhood that is a CAT(κ) space. For example, every Riemannian manifold with sectional curvature bounded above by κ is a CBA(κ) space.

▶ **Notation 6.** Let M be a compact CBA(κ) space. We denote $\rho_M^\kappa := \min\{\varrho_M, \frac{\varpi_\kappa}{4}\} > 0$.

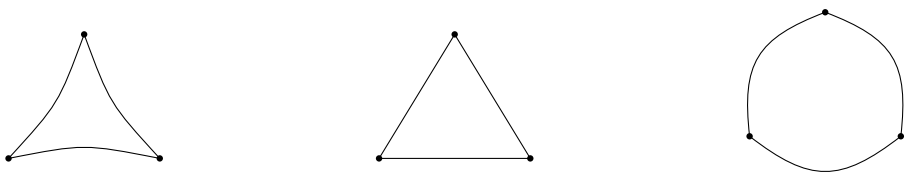


Figure 2 Illustration of equilateral triangles in curvatures $\kappa < 0$, $\kappa = 0$, and $\kappa > 0$, respectively.

2.2 Models for persistent homotopy types

We will use three models for *persistent homotopy types* in the following, arising from spaces, simplicial complexes, and simplicial sets, the last of which we discuss in Section 2.4.

▶ **Notation 7.** We denote by **sCplx** the category of simplicial complexes and simplicial maps, and by **Top** the category of topological spaces and continuous maps. We denote by $|K|$ the topological realization of a simplicial complex K. Often, we will just treat a complex as a space, and leave the realization implicit. The standard n-simplex, given by the set of non-empty subsets of $\{0,\dots,n\}$, will be denoted Δ^n_c.

▶ **Definition 8.** In the following, $\mathbb{P}$ will always denote a partially ordered set. When we treat $\mathbb{P}$ as a category, we mean the category whose objects are given by the elements of $\mathbb{P}$, and where there is a unique morphism $x \to y$ whenever $x \leq y$.
Let $\mathbf{C}$ be a category. A $\mathbb{P}$-*persistent object* in $\mathbf{C}$ is a functor $F\colon \mathbb{P} \to \mathbf{C}$. A morphism of $\mathbb{P}$-persistent objects is a natural transformation between such functors. We denote by $\mathbf{C}^{\mathbb{P}}$ the category of $\mathbb{P}$-persistent objects in $\mathbf{C}$ and their morphisms.

▶ **Notation 9.** We will usually denote persistent objects $F\colon \mathbb{P} \to \mathbf{C}$ in the form $F^\bullet$, to indicate their functoriality in $\mathbb{P}$. When the objects in a category $\mathbf{C}$ have a specific name, such as simplicial complexes, topological spaces or metric spaces, we will often refer to $\mathbb{P}$-persistent objects in $\mathbf{C}$ by adding the prefix persistent to that name.

▶ **Notation 10.** $N \geq 0$ will always be a non-negative integer and $\mathbb{R}^N$ will be considered as a partially ordered set via componentwise comparison. By a *space function pair* (often just pair), we will mean a topological space X together with a function $f\colon X \to \mathbb{R}^N$. When we speak of a *metric pair*, this will mean that we equip X with a metric, inducing its topology.

▶ **Example 11.** Given a pair (X, f), the sublevel sets $X^u := f^{-1}\{v \in \mathbb{R}^N \mid v \leq u\}$, together with functoriality given by inclusions, gives rise to a persistent space $u \mapsto X^u$, which we denote by $X^\bullet$. We abuse notation here insofar as the construction evidently depends on f. As all structure maps are given by inclusions $X^\bullet$ is often also referred to as a *filtered space*.

▶ **Example 12** (function-Rips complex)**.** Given a metric space $\mathbb{M}$ and $\delta \geq 0$, the open Vietoris-Rips complex of $\mathbb{M}$ (only Rips complex henceforth), denoted $\mathcal{R}^\delta(\mathbb{M})$, is the simplicial complex whose set of simplices is given by

$$\{\{x_0,\dots,x_n\} \mid x_i \in \mathbb{M}, d(x_i,x_j) < \delta, \text{ for all } 0 \leq i,j \leq n\}.$$

Varying the parameter $\delta \geq 0$, one obtains a persistent simplicial complex $\mathcal{R}^\bullet(\mathbb{M})\colon \mathbb{R}_{\geq 0} \to \mathbf{sCplx}$, with functoriality on relations given by inclusions. Suppose now that $\mathbb{M}$ is additionally equipped with a (not-necessarily continuous) function $\mathbb{f}\colon \mathbb{M} \to \mathbb{R}^N$. Then, for every $u \in \mathbb{R}^N$, one can consider the Vietoris-Rips complex of $\mathbb{M}^u := \{x \in \mathbb{M} \mid \mathbb{f}(x) \leq u\}$. Varying u with δ fixed gives rise to a persistent simplicial complex $\mathcal{R}^\delta(\mathbb{M}^\bullet)\colon \mathbb{R}^N \to \mathbf{sCplx}$, $u \mapsto \mathcal{R}^\delta(\mathbb{M}^u)$, called the *function-Rips complex*. Varying both u and δ gives rise to the bivariate variant $\mathcal{R}^\bullet(\mathbb{M}^\bullet)\colon \mathbb{R}_{\geq 0} \times \mathbb{R}^N \to \mathbf{sCplx}$, $(u,\delta) \mapsto \mathcal{R}^\delta(\mathbb{M}^u)$, called the *bivariate function-Rips complex*.

2.3 Persistent homotopy theory

Conceptually speaking, a $\mathbb{P}$-persistent homotopy type should be a $\mathbb{P}$-indexed functor valued in the *homotopy theory* or ∞-*category* of spaces, not just the ordinary 1-category of spaces (see also [21, 8]). As we do not expect familiarity with ∞-categorical language, we will instead work with the more elementary notion of relative categories, which we refer to as homotopy theories here.[1]

[1] This is justified insofar as relative categories provide a model for $(\infty,1)$-categories [28, 6, 7].

▶ **Definition 13.** By a *homotopy theory* we will mean a relative category; that is, a pair consisting of a category $\mathbf{C}$ and a wide subcategory $W \subset \mathbf{C}$. The morphisms in W are called *weak equivalences*. A morphism in $\mathbf{C}$ that is in W will often be denoted by the symbol $\xrightarrow{\simeq}$.

▶ **Example 14.** Recall that a weak homotopy equivalence between topological spaces is a continuous map that induces isomorphisms on the sets of path components and on all homotopy groups. The category **Top** of topological spaces equipped with the subcategory of weak homotopy equivalences forms a homotopy theory which we denote by $\mathcal{S}\mathbf{paces}$.[2]

▶ **Notation 15.** Given a homotopy theory $\mathcal{C} = (\mathbf{C}, W)$ and another category $\mathbb{I}$, we denote by $\mathcal{C}^{\mathbb{I}}$ the homotopy theory $(\mathbf{C}^{\mathbb{I}}, W^{\mathbb{I}})$, where $W^{\mathbb{I}}$ is the wide subcategory of $\mathbf{C}^{\mathbb{I}}$ consisting of those natural transformations $\varphi\colon F \Rightarrow G$ such that for every object $i \in \mathbb{I}$, the morphism $\varphi_i\colon F(i) \to G(i)$ is in W. [3] To indicate that we study functors (persistent objects) $F\colon \mathbb{I} \to \mathbf{C}$ in the context of the whole homotopy theory $\mathcal{C}$ we will often use the notation $F\colon \mathbb{I} \to \mathcal{C}$.

▶ **Example 16.** When $\mathcal{C} = \mathcal{S}\mathbf{paces}$ is the homotopy theory of spaces, then $\mathcal{S}\mathbf{paces}^{\mathbb{P}}$ will be referred to as the $\mathbb{P}$*-persistent homotopy theory of spaces*.

▶ **Notation 17.** Given a homotopy theory $\mathcal{C} = (\mathbf{C}, W)$, one can associate to it its homotopy category $\mathrm{ho}(\mathcal{C})$, defined as the localization $\mathbf{C}[W^{-1}]$ of $\mathbf{C}$ at the weak equivalences W.

▶ **Recollection 18.** Together with the canonical localization functor, $\mathbf{C} \to \mathbf{C}[W^{-1}] = \mathrm{ho}(\mathcal{C})$, $\mathrm{ho}(\mathcal{C})$ is characterized by the universal property that any functor $\mathbf{C} \to \mathbf{D}$ that sends weak equivalences to isomorphisms factors uniquely through $\mathrm{ho}(\mathcal{C})$. Explicitly, objects of $\mathrm{ho}(\mathcal{C})$ are the same as those of $\mathbf{C}$, and morphisms are given by equivalence classes of zig-zags of morphisms in $\mathbf{C}$, where backward-pointing arrows are weak equivalences. In this sense, $\mathrm{ho}(\mathcal{C})$ is the 1-category obtained by formally inverting the weak equivalences.

▶ **Notation 19.** We abuse notation insofar as we use the same symbols for objects and morphism in $\mathbf{C}$ and their images in $\mathrm{ho}(\mathcal{C})$. In the context of the persistent homotopy theory $\mathcal{S}\mathbf{paces}^{\mathbb{P}}$, a persistent space or persistent simplicial complex $X^\bullet$ will often be referred to as a *persistent homotopy type*. Note that while, set theoretically speaking, the persistent homotopy type $X^\bullet$ is the same as its underlying persistent space, in the context of the homotopy category $\mathrm{ho}(\mathcal{S}\mathbf{paces}^{\mathbb{U}})$, two persistent homotopy types are isomorphic when they are connected by a zig-zag of weak equivalences.

2.4 Persistent simplicial sets

At least half of the persistent homotopy types we are studying in this article arise from purely combinatorial data provided in the form of persistent simplicial complexes. It is thus convenient to have a model for the persistent homotopy theory of spaces that is more combinatorial in nature. This can be achieved by working with simplicial sets[4]. The reader not familiar with the theory of simplicial sets can treat them as a black-box that extends the category of simplicial complexes in a convenient way (see [20, 21] for an introduction).

---

[2] It follows as a consequence of Whitehead's theorem that this homotopy theory is equivalent to the one given by CW-complexes and homotopy equivalences.

[3] Note that this will generally only produce the ∞-categorical functor category when $\mathcal{C}$ is sufficiently well-behaved, for example when it extends to a model category, as is the case for our examples ([28]).

[4] From a categorical perspective, simplicial sets are a much preferable combinatorial model than simplicial complexes. Note, for example, that simplicial sets come with a natural weak equivalence $\mathrm{sd}_b X \xrightarrow{\simeq} X$ between the barycentric subdivision $\mathrm{sd}_b X$ of X and X, which we use in Section 5.

▶ **Recollection 20.** Recall that, conceptually speaking, a simplicial set is like a simplicial complex, where the simplices are ordered, and one allows for faces of simplices to collapse to lower dimensions. Categorically, this idea can be formalized as follows: Denote by Δ the category of finite linear posets $[n] = \{0 \leq \cdots \leq n\}$, for $n \geq 0$, with order-preserving maps. The category of simplicial sets $\mathbf{sSet}$ is the category of functors from Δ^{op} into $\mathbf{Set}$, i.e., $\mathbf{Set}^{\Delta^{\mathrm{op}}}$. Given a simplicial set $X \in \mathbf{sSet}$ and $n \geq 0$, the set $X([n])$ is denoted X_n, and called the set of n-simplices of X. A simplex $\sigma \in X_n$ that is not in the image of a structure map $X_k \to X_n$, for some $k < n$, is called non-degenerate.

▶ **Example 21.** As a simplicial set, the function-Rips complex at $u \in \mathbb{R}^N$ and $\delta \geq 0$ can be modeled by a simplicial set with n-simplices

$$\mathcal{R}^{u,\delta}(M,f)_n = \{(x_0,\ldots,x_n) \mid x_i \in M; d(x_i,x_j) < \delta; f(x_i) \leq u \text{ for all } i,j \in [n]\},$$

and functoriality on Δ^{op} given by precomposition.

▶ **Notation 22.** Given $n \in \mathbb{N}$, the image of $[n]$ under the Yoneda embedding $\Delta \hookrightarrow \mathbf{Set}^{\Delta^{\mathrm{op}}}$, $[k] \mapsto \Delta([k],[n])$, is denoted by Δ^n and referred to as the n-simplex. The functor $[n] \mapsto \Delta^n$ defines a fully faithful embedding $\Delta \hookrightarrow \mathbf{sSet}$, by which we treat Δ as a subcategory of $\mathbf{sSet}$.

▶ **Recollection 23.** Simplicial sets admit a topological realization functor. Observe that Δ embeds into $\mathbf{sCplx}$ by sending $[n]$ to the standard simplex Δ^n_c. The composition $\Delta \to \mathbf{sCplx} \xrightarrow{|-|} \mathbf{Top}$ defines a topological realization functor on $\Delta \subset \mathbf{sSet}$, which extends canonically to a colimit-preserving functor $|-|\colon \mathbf{sSet} \to \mathbf{Top}$, so that the realization of a simplicial set X, $|X|$, is glued from realizations of its simplices.

The most important reason why simplicial sets are so useful is that they can be used to define a homotopy theory equivalent to that of topological spaces.

▶ **Recollection 24.** A simplicial map $\varphi\colon X \to Y$ in $\mathbf{sSet}$ is called a weak homotopy equivalence if its topological realization $|\varphi|\colon |X| \to |Y|$ is a homotopy equivalence. Denote the wide subcategory of weak homotopy equivalences by W_{Kan}, and the resulting relative category $(\mathbf{sSet}, W_{\mathrm{Kan}})$ by $\mathbf{sSet}_{\mathrm{Kan}}$. It is a fundamental fact of homotopy theory, that the topological realization functor $|-|\colon \mathbf{sSet} \to \mathbf{Top}$ then defines a so-called *equivalence of homotopy theories (∞-categories)* $\mathbf{sSet}_{\mathrm{Kan}} \to \mathcal{S}\mathbf{paces}$ (see [32]). Conceptually speaking, this means that any homotopy theoretic construction or argument concerning $\mathcal{S}\mathbf{paces}$ can equivalently be performed in $\mathbf{sSet}_{\mathrm{Kan}}$. For our purposes, however, it suffices to observe that the functor $|-|\colon \mathbf{sSet}^{\mathbb{P}} \to \mathbf{Top}^{\mathbb{P}}$ descends to an equivalence of categories $|-|\colon \mathrm{ho}\big((\mathbf{sSet}, W_{\mathrm{Kan}})^{\mathbb{P}}\big) \to \mathrm{ho}(\mathcal{S}\mathbf{paces}^{\mathbb{P}})$, compatible with any reparametrization of the indexing poset $\mathbb{P}$.

▶ **Recollection 25.** In the same way as homotopies induce identifications of maps in the homotopy category $\mathrm{ho}(\mathcal{S}\mathbf{paces})$, morphisms of persistent simplicial sets of the form $X^\bullet \times \Delta^1 \to Y^\bullet$ (where $X^\bullet \times \Delta^1$ denotes the indexwise product) – so-called *elementary homotopies* – induce identifications of persistent simplicial maps in $\mathrm{ho}(\mathbf{sSet}^{\mathbb{P}}_{\mathrm{Kan}})$.

▶ **Recollection 26.** We frequently want to treat simplicial complexes as simplicial sets. There is a canonical fully faithful embedding $N_s\colon \mathbf{sCplx} \hookrightarrow \mathbf{sSet}$, mapping a simplicial complex, K, to the simplicial set given by $N_s(K)_n = \mathbf{sCplx}(\Delta^n_c, K)$, functorial in the obvious way in n and K (see [16, § 1.4]). For example, the simplicial set obtained by applying N_s to a function-Rips complex is precisely the one described in Example 21. We will usually omit N_s from the notation. Observe that from a homotopy-theoretic perspective, this is justified by the fact that, given $K \in \mathbf{sCplx}$, there is a canonical homotopy equivalence $|N_s(K)| \xrightarrow{\simeq} |K|$ (see [30, 5]) even though the equality $|K| \cong |N_s(K)|$ does not hold on the homeomorphism level. Hence, for our purposes, we can freely identify the two realizations.

2.5 Interleavings in the homotopy category

One of the core advantages of persistent settings is that they allow for approximate notions of equivalence, so called interleavings (see [17] for a more general setting). In the following, $\mathcal{C}$ will denote some homotopy theory. To define interleavings, we need the following notation.

▶ **Notation 27.** Henceforth, $\mathbb{U} \subset \mathbb{R}^N$, $\mathbb{U}' \subset \mathbb{R}^{N'}$ will always denote upsets of $\mathbb{R}^N$, i.e., subsets fulfilling $x \in \mathbb{U}, x \leq y \implies y \in \mathbb{U}$.
Suppose we are given a map of posets $S\colon \mathbb{U}' \to \mathbb{U}$ and a persistent object $F\colon \mathbb{U} \to \mathcal{C}$. We write $F^{S(\bullet)}$ to denote the persistent object obtained by precomposing F with S. In the special case where S is an inclusion of posets, we use the notation $F^\bullet|_{\mathbb{U}'}$ to denote the restriction of F to $\mathbb{U}'$. Suppose we are given another such map $S'\colon \mathbb{U}' \to \mathbb{U}$ such that $S(x) \leq S'(x)$, for all $x \in \mathbb{U}'$. In this case, we will use the notation $s\colon F^{S(\bullet)} \to F^{S'(\bullet)}$ to denote the natural transformation induced by the relations $S(x) \leq S'(x)$. Given $\varepsilon \geq 0$ write $(-) + \varepsilon\colon \mathbb{U} \to \mathbb{U}$ for the map of posets given by $x \mapsto x + \varepsilon(1, \dots, 1)$.

▶ **Definition 28.** An *ε-interleaving in the homotopy category* between $F^\bullet, G^\bullet\colon \mathbb{U} \to \mathcal{C}$ consists of morphisms $\varphi\colon F^\bullet \to G^{\bullet+\varepsilon}$ and $\psi\colon G^\bullet \to F^{\bullet+\varepsilon}$ in $\mathrm{ho}(\mathcal{C}^{\mathbb{U}})$ such that the diagrams

$$\begin{array}{ccc} F^\bullet & \xrightarrow{\quad s \quad} & F^{\bullet+2\varepsilon} \\ & {\scriptstyle\varphi}\searrow \quad \nearrow{\scriptstyle\psi^{+\varepsilon}} & \\ & G^{\bullet+\varepsilon} & \end{array} \qquad \begin{array}{ccc} G^\bullet & \xrightarrow{\quad s \quad} & G^{\bullet+2\varepsilon} \\ & {\scriptstyle\psi}\searrow \quad \nearrow{\scriptstyle\varphi^{+\varepsilon}} & \\ & F^{\bullet+\varepsilon} & \end{array} \tag{1}$$

in $\mathrm{ho}(\mathcal{C}^{\mathbb{U}})$ commute. We will denote such interleavings in the form $\varphi\colon F \simeq_\varepsilon G\colon \psi$.

▶ Remark 29. Two interleavings $\varphi\colon F^\bullet \simeq_\varepsilon G^\bullet\colon \psi$ and $\varphi'\colon G^\bullet \simeq_{\varepsilon'} J^\bullet\colon \psi'$ compose to an interleaving $(\varphi')^{+\varepsilon} \circ \varphi\colon F^\bullet \simeq_{\varepsilon+\varepsilon'} J^\bullet\colon \psi^{+\varepsilon'} \circ \psi'$ (see, for example, [17]). It follows that $d_{IH}(F^\bullet, G^\bullet) := \inf\{\varepsilon \geq 0 \mid F^\bullet \simeq_\varepsilon G^\bullet\}$, defines an extended pseudo distance on $\mathcal{C}^{\mathbb{U}}$ called the *interleaving distance in the homotopy category* (see [24]). Observe, also, that any interleaving $\varphi\colon F^\bullet \simeq_\varepsilon G^\bullet\colon \psi$ gives rise to an interleaving $s \circ \varphi\colon F^\bullet \simeq_{\varepsilon''} G^\bullet\colon s \circ \psi$, for any $\varepsilon'' \geq \varepsilon$.

▶ **Example 30.** When $\mathbb{U} = \mathbb{R}^0 = \{0\}$, we can identify $\mathcal{C}^{\mathbb{U}} = \mathcal{C}$, and $(-) + \varepsilon\colon \mathbb{U} \to \mathbb{U}$ is the identity. Then, an ε-interleaving in $\mathrm{ho}(\mathcal{C})$ is the same as an isomorphism in $\mathrm{ho}(\mathcal{C})$.

3 Stability of the bivariate function-Rips persistent homotopy type

To study the stability properties of the assignment $(\mathbb{M}, \mathbb{f}) \mapsto \mathcal{R}^\delta(\mathbb{M}^\bullet)$, it is useful to consider the latter as a bivariate construction by also varying δ.

▶ **Notation 31.** Given a persistent object of the form $F\colon \mathbb{R}_{\geq 0} \times \mathbb{U} \to \mathbf{C}$, we will use the notation $F^{\bullet,\bullet}$ to indicate the fact that it is a functor in two variables.

▶ **Definition 32** (See also [11])**.** Let $(\mathbb{M}_0, \mathbb{f}_0)$ and $(\mathbb{M}_1, \mathbb{f}_1)$ be metric pairs over $\mathbb{R}^N$, and let $\varepsilon \geq 0$. A *filtered ε-correspondence* between $(\mathbb{M}_0, \mathbb{f}_0)$ and $(\mathbb{M}_1, \mathbb{f}_1)$ is a subset $\mathfrak{C} \subset \mathbb{M}_0 \times \mathbb{M}_1$ such that $\pi_{\mathbb{M}_0}(\mathfrak{C}) = \mathbb{M}_0$ and $\pi_{\mathbb{M}_1}(\mathfrak{C}) = \mathbb{M}_1$, and such that, for all $(x, y), (x', y') \in \mathfrak{C}$, it holds that

$$|d_{\mathbb{M}_0}(x, x') - d_{\mathbb{M}_1}(y, y')| \leq \varepsilon \text{ and } |\mathbb{f}_0(x) - \mathbb{f}_1(y)|_\infty \leq \varepsilon.$$

We will denote filtered ε-correspondences in the form $\mathfrak{C}\colon (\mathbb{M}_0, \mathbb{f}_0) \approx_\varepsilon (\mathbb{M}_1, \mathbb{f}_1)$.

▶ Remark 33. Filtered correspondences give rise to a distance on metric pairs $(\mathbb{M}_0, \mathbb{f}_0)$ and $(\mathbb{M}_1, \mathbb{f}_1)$, defined as the infimum of the values ε such that there exists a filtered ε-correspondence $(\mathbb{M}_0, \mathbb{f}_0) \approx_\varepsilon (\mathbb{M}_1, \mathbb{f}_1)$. When $N = 0$, this distance is exactly twice the ordinary Gromov-Hausdorff distance.

The additional flexibility of varying the Rips parameter δ allows for a proof of the following stability result, homological variants of which can be found in several places ([12, 8]).

▶ **Proposition 34.** *A filtered ε-correspondence $\mathfrak{C}\colon (\mathbb{M}_0, \mathbb{f}_0) \approx_\varepsilon (\mathbb{M}_1, \mathbb{f}_1)$ gives rise to an interleaving $\mathcal{R}^\bullet(\mathbb{M}_0^\bullet) \simeq_\varepsilon \mathcal{R}^\bullet(\mathbb{M}_1^\bullet)$ in $\mathrm{ho}(\mathcal{S}\mathbf{paces}^{\mathbb{R}_{\geq 0}\times\mathbb{R}^N})$.*

We use Recollections 24 and 26 and construct the interleaving using the simplicial set model for the bivariate Rips complex. We will leverage a universal property of the Rips simplicial set here, which we state for the case of filtered simplicial sets $X^{\bullet,\bullet} \in \mathbf{sSet}^{\mathbb{R}_{\geq 0}\times\mathbb{R}^N}$, i.e., where all structure morphisms are inclusions (see Proposition 36 below.) It allows one to construct morphisms $X^{\bullet,\bullet} \to \mathcal{R}^\bullet(\mathbb{M}^\bullet)$ by defining a map with target $\mathbb{M}$ on the vertices of $X^{\bullet,\bullet}$ (see [13] which uses the simplicial complex analog). We write $X^\infty = \bigcup_{(\delta,u)\in\mathbb{R}_{\geq 0}\times\mathbb{R}^N} X^{\delta,u}$.

▶ **Construction 35.** Let $X\colon \mathbb{R}_{\geq 0}\times\mathbb{R}^N \to \mathbf{sSet}$ be a filtered simplicial set and $(\mathbb{M}, \mathbb{f})$ be a metric pair over $\mathbb{R}^N$. We denote by $S(X, (\mathbb{M}, \mathbb{f})) \subset \mathbf{Set}(X_0^\infty, \mathbb{M})$ the set of such maps $\varphi\colon X_0^\infty \to \mathbb{M}$ that fulfill the condition that, for all $(\delta, u) \in \mathbb{R}_{\geq 0}\times\mathbb{R}^N$, it holds that:
1. for all $x \in X_0^{\delta,u}$, we have $\mathbb{f}(\varphi(x)) \leq u$;
2. for all 1-simplices $\sigma \in X_1^{\delta,u}$, with vertices x_0 and x_1, we have $d(\varphi(x_0), \varphi(x_1)) < \delta$.

Associating to a morphism $X^{\bullet,\bullet} \to \mathcal{R}^\bullet(\mathbb{M}^\bullet)$ the induced simplicial map $X^\infty \to \mathcal{R}^\infty(\mathbb{M}, \mathbb{f}) = \mathcal{R}^\infty(\mathbb{M})$ and evaluating on vertices to obtain a map $X_0^\infty \to \mathcal{R}^\infty(\mathbb{M})_0 = \mathbb{M}$, defines a map

$$\eta\colon \mathbf{sSet}^{\mathbb{R}_{\geq 0}\times\mathbb{R}^N}(X^{\bullet,\bullet}, \mathcal{R}^\bullet(\mathbb{M}^\bullet)) \to S(X, (\mathbb{M}, \mathbb{f})).$$

▶ **Proposition 36** (See full version [31])**.** *η, as in Construction 35, defines a natural bijection.*

▶ Remark 37. Conceptually speaking, the injectivity part of Proposition 36 means that a morphism $X^{\bullet,\bullet} \to \mathcal{R}^\bullet(\mathbb{M}^\bullet)$ is uniquely determined by its values on vertices (the values can be identified with elements of $\mathbb{M}$). The surjectivity and well-definedness state that a map $X_0^\infty \to \mathbb{M}$ extends if and only if the criteria of Construction 35 are verified.

▶ Remark 38. We will often need to use a modified version of Proposition 36, where instead of $\mathcal{R}^\bullet(\mathbb{M}^\bullet)$, we consider a reparametrized version in which $\mathcal{R}^\bullet(\mathbb{M}^\bullet)\colon \mathbb{R}_{\geq 0}\times\mathbb{R}^N \to \mathbf{sSet}$ is replaced by a composition $\mathbb{R}_{\geq 0}\times\mathbb{R}^N \xrightarrow{(\delta',u')} \mathbb{R}_{\geq 0}\times\mathbb{R}^N \xrightarrow{\mathcal{R}^\bullet(\mathbb{M}^\bullet)} \mathbf{sSet}$ for poset maps δ', u'. Then, the analogous statement holds, replacing in Construction 35 u with $u'(\delta, u)$ and δ with $\delta'(\delta, u)$ in the defining inequalities for $S(X^{\bullet,\bullet}, (\mathbb{M}, \mathbb{f}))$. Furthermore, we will often encounter the modified case where we work with filtered simplicial sets $X^{\bullet,\bullet}$ over $\mathbb{I}\times\mathbb{R}^N$, with $\mathbb{I} = [0, \rho]$ or $\mathbb{I} = \{\rho\}$, and we restrict $\mathcal{R}^\bullet(\mathbb{M}^\bullet)$ to $\mathbb{I}\times\mathbb{R}^N$. In this case, again the essentially same statement holds, with the only change being the replacement of X^∞ by $X^{\rho,\infty} = \bigcup_{u\in\mathbb{R}^N} X^{\rho,u}$ and the defining condition for $S(X^{\bullet,\bullet}, (\mathbb{M}, \mathbb{f}))$ needing to be verified only for pairs $(\delta, u) \in \mathbb{I}\times\mathbb{R}^N$.

Next, let us construct the simplicial maps that define the interleaving of Proposition 34:

▶ **Construction 39.** Given a filtered ε-correspondence $\mathfrak{C}$, we define maps

$$\varphi\colon \mathcal{R}^\bullet(\mathbb{M}_0^\bullet) \to \mathcal{R}^{\bullet+\varepsilon}(\mathbb{M}_1^{\bullet+\varepsilon}) \quad \text{and} \quad \psi\colon \mathcal{R}^\bullet(\mathbb{M}_1^\bullet) \to \mathcal{R}^{\bullet+\varepsilon}(\mathbb{M}_0^{\bullet+\varepsilon})$$

by applying Proposition 36 as follows. Observe that for $X^{\bullet,\bullet} = \mathcal{R}^\bullet(\mathbb{M}_0^\bullet)$, we have $X_0^\infty = \mathbb{M}_0$. Then φ is defined under Proposition 36 by choosing for each $x \in \mathbb{M}_0$ an element $y \in \mathbb{M}_1$ with $(x, y) \in \mathfrak{C}$ and defining $\varphi(x) := y$. The map ψ is defined analogously.

Proof of Proposition 34. The conditions on $\mathfrak{C}$ ensure exactly that the (shifted) requirements of Proposition 36 and Remark 38 are fulfilled. It remains to see that φ and ψ define an ε-interleaving in the homotopy category. We show that $\psi^{+\varepsilon} \circ \varphi \simeq s$, the other equality

being shown analogously. To this end, we need to construct a homotopy $H\colon \mathcal{R}^{\bullet}(\mathbb{M}_0^{\bullet}) \times \Delta^1 \to \mathcal{R}^{\bullet+2\varepsilon}(\mathbb{M}_0^{\bullet+2\varepsilon})$ between $\psi^{+\varepsilon} \circ \varphi$ and s. We again use Proposition 36. In this case, it implies that the persistent simplicial maps $\psi^{+\varepsilon} \circ \varphi$ and s are elementarily homotopic, if for every pair $x, y \in \mathbb{M}_0$ it holds that $d(\psi^{+\varepsilon}(\varphi(x)), s(y)) \leq d(x,y) + 2\varepsilon$. Indeed, the inequality

$$d(\psi^{+\varepsilon}(\varphi(x)), s(y)) = d(\psi^{+\varepsilon}(\varphi(x)), y) \leq d(\varphi(x), \varphi(y)) + \varepsilon \leq d(x,y) + 2\varepsilon,$$

holds by definition of φ and ψ in terms of $\mathfrak{C}$. ◀

4 The shrinking trick

For the remainder of this section, we will fix some homotopy theory $\mathcal{C}$. By an interleaving, we will always mean interleaving in the homotopy category.

▶ **Notation 40.** When fixing either parameter of a persistent object $F^{\bullet,\bullet}\colon \mathbb{R}_{\geq 0} \times \mathbb{U} \to \mathcal{C}$, we will use the notation $F^{\delta,\bullet}$ or $F^{\bullet,u}$, to denote the resulting persistent objects on $\mathbb{U}$, or $\mathbb{R}_{\geq 0}$ respectively. At times, we will also use the notation $\bullet_1$ and $\bullet_2$ to indicate the first and second variable, respectively. For example $F^{\bullet_1,\bullet_2+\bullet_1}$ indicates the persistence module given by precomposing F with the endofunctor $(\delta, u) \mapsto (\delta, u+\delta)$ of $\mathbb{R}_{\geq 0} \times \mathbb{U}$.

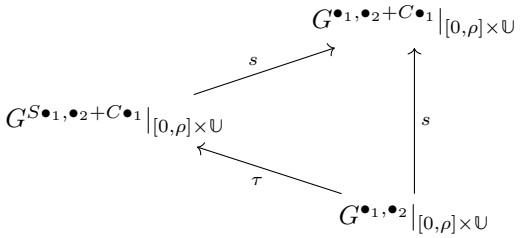
Suppose we are given $\delta \geq 0$, as well as two persistent objects $F^{\bullet,\bullet}, G^{\bullet,\bullet}\colon \mathbb{R}_{\geq 0} \times \mathbb{U} \to \mathcal{C}$, together with an ε-interleaving $\varphi\colon F^{\bullet,\bullet} \simeq_\varepsilon G^{\bullet,\bullet}\colon \psi$. We want to deduce an interleaving $F^{\delta,\bullet} \simeq_{\varepsilon'} G^{\delta,\bullet}$, for some ε' depending on ε and δ. The issue at hand is, of course, that the interleaving morphisms φ and ψ only procure morphisms $F^{\delta,\bullet} \to G^{\delta+\varepsilon,\bullet+\varepsilon}$ and $G^{\delta,\bullet} \to F^{\delta+\varepsilon,\bullet+\varepsilon}$. To amend this, we introduce an additional structure on G, which we call a shrinking transformation. This transformation allows us to decrease the δ-part of the persistence parameter of G, at the cost of increasing the u-part.

▶ **Notation 41.** We fix constants $0 < S < 1, C \geq 0$ and $\rho > 0$ in $\mathbb{R}$.

▶ **Definition 42.** A *shrinking transformation* for $G\colon \mathbb{R}_{\geq 0} \times \mathbb{U} \to \mathcal{C}$ is a morphism

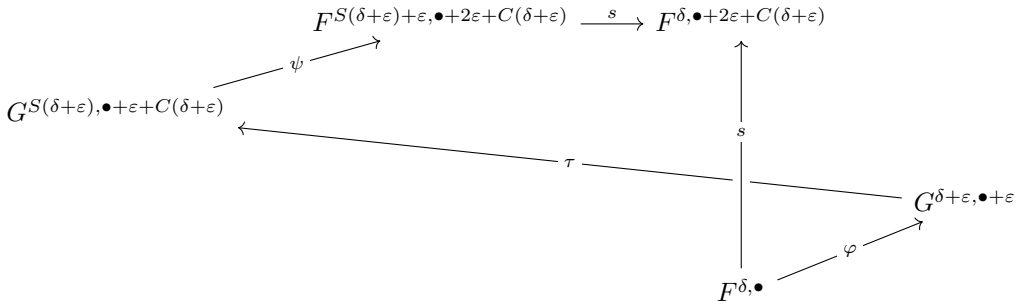
$$\tau\colon G^{\bullet_1,\bullet_2}|_{[0,\rho]\times\mathbb{U}} \to G^{S\bullet_1,\bullet_2+C\bullet_1}|_{[0,\rho]\times\mathbb{U}}$$

in $\mathrm{ho}(\mathcal{C}^{[0,\rho]\times\mathbb{U}})$ such that the following diagram in $\mathrm{ho}(\mathcal{C}^{[0,\rho]\times\mathbb{U}})$ commutes:



(2)

▶ **Definition 43.** Let $\delta \in [0,\rho]$ and $0 \leq \varepsilon$ be such that $\varepsilon \leq \min\{\rho - \delta, \frac{1-S}{1+S}\delta\}$. As we assumed $\varepsilon + \delta \leq \rho$, the morphism τ from Definition 42 restricts to a well-defined morphism $\tau\colon G^{\delta+\varepsilon,\bullet+\varepsilon} \to G^{S(\delta+\varepsilon),\bullet+\varepsilon+C(\delta+\varepsilon)}$, called by the same name by abuse of notation. We say that an ε-interleaving $\varphi\colon F^{\bullet,\bullet} \simeq_\varepsilon G^{\bullet,\bullet}\colon \psi$ is δ-*compatible* with τ if the following diagram in $\mathrm{ho}(\mathcal{C}^{\mathbb{U}})$ commutes:



(3)

▶ **Theorem 44.** *Let $G\colon \mathbb{R}_{\geq 0} \times \mathbb{U} \to \mathcal{C}$ be a persistent object in $\mathcal{C}$, equipped with a shrinking transformation τ on G. Let $\varepsilon, \delta \geq 0$ be such that $\varepsilon \leq \min\{\rho - \delta, \frac{1-S}{1+S}\delta\}$.*
Furthermore, let $F\colon \mathbb{R}_{\geq 0} \times \mathbb{U} \to \mathcal{C}$ be another persistent object, together with an ε-interleaving $\varphi\colon F \simeq_\varepsilon G\colon \psi$ that is δ-compatible with τ. Then there is an interleaving

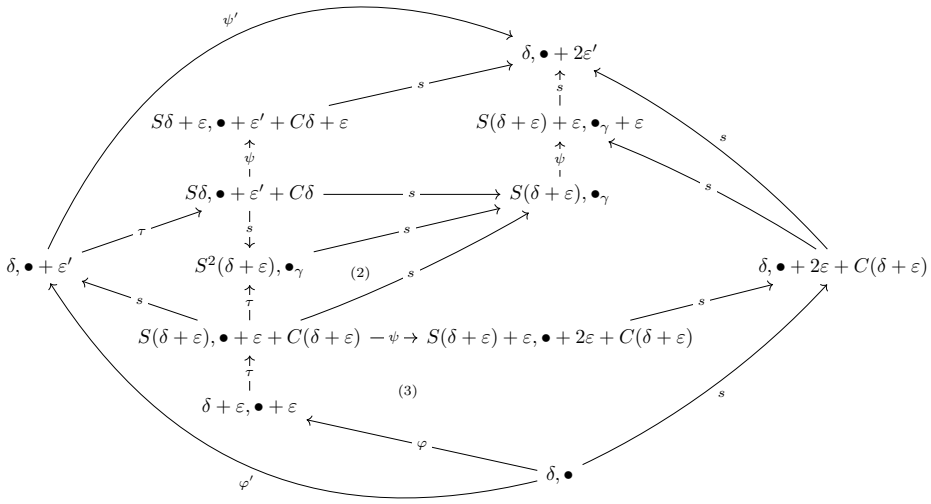
$$\varphi'\colon F^{\delta,\bullet} \simeq_{(1+C)\varepsilon+C\delta} G^{\delta,\bullet}\colon \psi',$$

explicitly given by the compositions

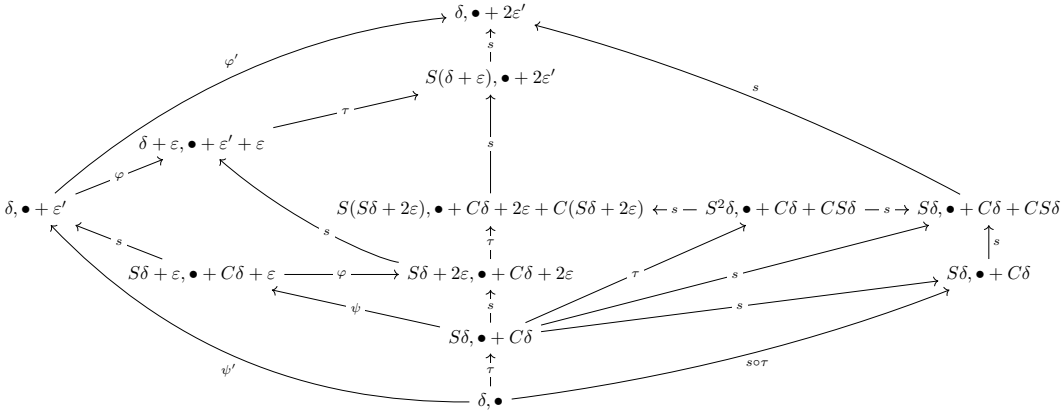
$$\varphi'\colon F^{\delta,\bullet} \xrightarrow{\varphi} G^{\delta+\varepsilon,\,\bullet+\varepsilon} \xrightarrow{\tau} G^{S(\delta+\varepsilon),\,\bullet+(1+C)\varepsilon+C\delta} \xrightarrow{s} G^{\delta,\,\bullet+(1+C)\varepsilon+C\delta},$$

$$\psi'\colon G^{\delta,\bullet} \xrightarrow{\tau} G^{S\delta,\,\bullet+C\delta} \xrightarrow{\psi} F^{S\delta+\varepsilon,\,\bullet+C\delta+\varepsilon} \xrightarrow{s} F^{\delta,\,\bullet+(1+C)\varepsilon+C\delta}.$$

Proof. The fact that all morphisms s and τ in the definition of φ' and ψ' are well-defined follows from the inequalities $\delta + \varepsilon \leq \rho$ and $S(\delta + \varepsilon) \leq S\delta + \varepsilon \leq S(\delta + \varepsilon) + \varepsilon \leq \delta$, which hold by the assumption on ε and δ. We now need to verify the defining commutativity conditions of an interleaving. To simplify notation, we define $\varepsilon' := (1 + C)\varepsilon + C\delta$ and $\bullet_\gamma := \bullet_2 + \varepsilon + C(\delta + \varepsilon) + CS(\delta + \varepsilon)$. To simplify notation even further, we will omit F and G from the notation and only spell out the superscript. Whether F or G is meant will be uniquely determined by the specified morphisms. Now, consider the first composition $\psi' \circ \varphi'$. To verify that $\psi' \circ \varphi' = s$ it suffices to verify the commutativity of the following diagram.



Observe that for every s arrow to be well-defined, we require the inequality $S(\delta+\varepsilon)+\varepsilon \leq \delta$, which we have already seen above. Observe that the canonical shift morphisms s commute with essentially every other morphism in sight (in the appropriate contextual sense). The commutativities of the cells in the diagram follow either from the universal commutativity of s or from Diagrams (2) and (3). This proves the first interleaving equality.
The second interleaving equality follows by chasing the following diagram.



Note that none of the cells require the δ-compatibility condition. Instead, one only uses the naturality properties of s, one of the two interleaving equalities, and the defining property of the shrinking transformation. ◀

5 Shrinking transformations for the bivariate function-Rips complex

We now want to apply Theorem 44 together with Proposition 34 to prove Theorem 3.

▶ **Notation 45.** For the remainder of this section, let M be a CBA(κ) space, for some $\kappa \in \mathbb{R}$, equipped with a 1-Lipschitz function $f\colon M \to \mathbb{R}^N$ (with respect to the $|-|_\infty$-norm on $\mathbb{R}^N$), and let S, C and ρ be as in Notation 41.

To establish the existence of a shrinking morphism, we use a classical technique in simplicial homotopy theory: Leveraging subdivisions to construct maps in the homotopy category.

▶ **Notation 46.** For the remainder of this section, we fix what we call a *subdivision functor*, by which we mean a pair consisting of a functor $\mathrm{sd}\colon \mathbf{sSet}_{\mathrm{Kan}} \to \mathbf{sSet}_{\mathrm{Kan}}$ that preserves inclusions and colimits, together with a natural weak equivalence $\lambda\colon \mathrm{sd} \xRightarrow{\simeq} 1_{\mathbf{sSet}}$.

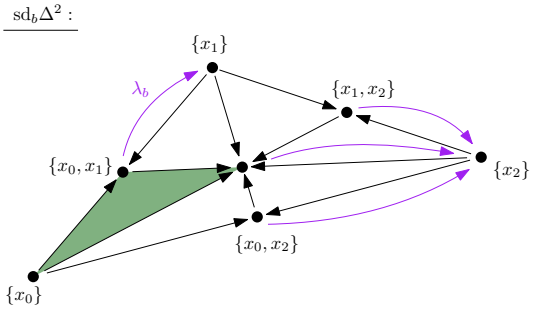


Figure 3 Illustration of the last vertex map λ_b on $\mathrm{sd}_b\Delta^2$. The purple arrows indicate where the vertices are mapped by λ_b. All but the green simplex are collapsed to lower-dimensional simplices.

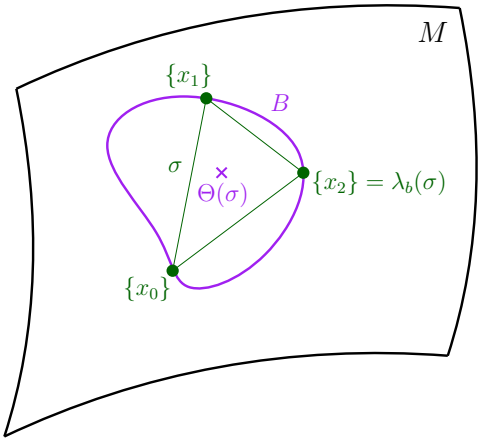


Figure 4 Illustration of the pseudo-barycenter map Θ.

▶ **Example 47.** For the purpose of this article, the only relevant example is the *barycentric subdivision functor* $\mathrm{sd}_b\colon \mathbf{sSet} \to \mathbf{sSet}$ together with the last vertex map $\lambda_b\colon \mathrm{sd}_b X \xrightarrow{\simeq} X$ (see [22]). Let us recall the special case $X = \mathcal{R}^\delta(M)$, which is the only one we will need. When $X = \mathcal{R}^\delta(M)$, each vertex of $\mathrm{sd}_b\mathcal{R}^\delta(M)$ corresponds to a sequence $(x_0, \dots, x_n)$ of elements of M that are less than δ apart from one another, and such that $x_i \neq x_{i+1}$ for all $i \in [n-1]$. Then. $\lambda_b\colon \mathrm{sd}_b\mathcal{R}^\delta(M) \xrightarrow{\simeq} \mathcal{R}^\delta(M)$ is given on vertices by mapping $(x_0, \dots, x_n) \in \mathrm{sd}_b(\mathcal{R}^\delta(M))_0$ to $x_n \in M = \mathcal{R}^\delta(M)_0$.

Shrinking transformations will then be obtained from the following types of maps:

▶ **Definition 48.** By a *pseudo-barycenter map*, we mean a map $\Theta\colon \mathrm{sd}\mathcal{R}^\rho(M)_0 \to M$ such that for all $0 < \delta \leq \rho$:

(a) For every $x \in \mathrm{sd}\mathcal{R}^\delta(M)_0$, we have $d(\lambda(x), \Theta(x)) \leq C\delta$;

(b) For $x, y \in \mathrm{sd}\mathcal{R}^\delta(M)_0$, if there is a 1-simplex from x to y in $\mathrm{sd}\mathcal{R}^\delta(M)$, then $d(\Theta(x), \Theta(y)) < S\delta$ and $d(\lambda(x), \Theta(y)) < S\delta$.

▶ **Example 49.** Let $\rho = \rho_M^\kappa$. Consider the map Θ that maps a vertex $\sigma = (x_0, \dots, x_n) \in \mathrm{sd}_b(\mathcal{R}^\rho(M))$ to the center of a minimal enclosing ball B of $\{x_0, \dots, x_n\} \subset M$- as illustrated in Figure 4. It follows from a geodesic version of Jung's theorem (see [23, Lemma 3.4], and also [34] for a related construction) that for any two finite sets $\{x_0, \dots, x_n\} \subset \{y_0, \dots, y_m\}$ of diameter $\delta < \rho_M^\kappa$, the associated centers $\Theta(x_0, \dots, x_n)$ and $\Theta(y_0, \dots, y_m)$ are well-defined and unique, and fulfill $d(\Theta(x_0, \dots, x_n), \Theta(y_0, \dots, y_m)) < \frac{3}{4}\delta$. As an immediate consequence, it follows that Θ defines a pseudo-barycenter map with constants $S = C = \frac{3}{4}$.

▶ **Construction 50.** We use Recollections 24 and 26 and construct the shrinking transformation in the homotopy category of persistent simplicial sets. Since sd preserves colimits, we have $\mathrm{sd}(\mathcal{R}^\bullet(M^\bullet))_0^{\infty,\rho} = \mathrm{sd}(\mathcal{R}^\rho(M))_0$. Now, under this identification, we can apply Proposition 36 and Remark 38 to uniquely extend Θ to a morphism $\tau'\colon \mathrm{sd}(\mathcal{R}^{\bullet_1}(M^{\bullet_2}))|_{[0,\rho]\times\mathbb{R}^N} \to \mathcal{R}^{S\bullet_1}(M^{\bullet_2 + C\bullet_1})|_{[0,\rho]\times\mathbb{R}^N}$ (see the proof of Theorem 51). Finally, in the persistent homotopy category $\mathrm{ho}(\mathbf{sSet}_{\mathrm{Kan}}^{[0,\rho]\times\mathbb{R}^N})$, we can invert the weak equivalence $\lambda\colon \mathrm{sd}\mathcal{R}^\bullet(M^\bullet) \simeq \mathcal{R}^\bullet(M^\bullet)$ and define τ_Θ as the composition:

$$\tau_\Theta\colon \mathcal{R}^{\bullet_1}(M^{\bullet_2})|_{[0,\rho]\times\mathbb{R}^N} \xrightarrow{\lambda^{-1}} \mathrm{sd}(\mathcal{R}^{\bullet_1}(M^{\bullet_2})|_{[0,\rho]\times\mathbb{R}^N}) \xrightarrow{\tau'} \mathcal{R}^{S\bullet_1}(M^{\bullet_2 + C\bullet_1})|_{[0,\rho]\times\mathbb{R}^N}.$$

Using this construction, we can state the main result of this section.

▶ **Theorem 51.** *The morphism* $\tau_\Theta\colon \mathcal{R}^{\bullet_1}(M^{\bullet_2})|_{[0,\rho]\times\mathbb{R}^N} \to \mathcal{R}^{S\bullet_1}(M^{\bullet_2+C\bullet_1})|_{[0,\rho]\times\mathbb{R}^N}$ *is a shrinking transformation and* δ*-compatible with every* ε*-interleaving* $\mathcal{R}^\bullet(\mathbb{M}^\bullet) \simeq_\varepsilon \mathcal{R}^\bullet(M^\bullet)$ *arising from a correspondence* $\mathfrak{C}\colon (\mathbb{M},\mathbb{f}) \approx_\varepsilon (M,f)$*, for* $\delta \le \rho$ *and* $\varepsilon \le \min\{\rho-\delta, \frac{1-S}{1+S}\delta\}$.

Proof. Throughout the proof, we are going to make use of the modified version of Proposition 36 in Remark 38. To simplify notation, we will write $\mathcal{R}^{\bullet,\bullet}$ for $\mathcal{R}^\bullet(M^\bullet)|_{[0,\rho]\times\mathbb{R}^N}$. We first show that τ', as defined in the previous construction, is indeed a well-defined morphism of persistent simplicial sets. Observe that compared to Proposition 36 the target persistent simplicial set has transformed under the parameter change $\bullet_1' = S\bullet_1$, $\bullet_2' = \bullet_2 + C\bullet_1$. The conditions that we need to verify for τ' to define a well-defined morphism (Remark 38) are explicitly given as follows. Let $\delta \in [0,\rho]$ and $u \in \mathbb{R}^N$. We need to verify the following:

1. Let $\sigma \in \mathrm{sd}\mathcal{R}^\delta(M)_1$ be a 1-simplex with vertices $x, y \in \mathrm{sd}\mathcal{R}^\delta(M)_0$. Then the inequality $d(\Theta(x),\Theta(y)) < S\delta$ holds. This condition is assumed by Property b.
2. For $x \in \mathrm{sd}\mathcal{R}_0^{u,\delta}$, it holds that $f(\Theta(x)) \le u + C\delta$. This is a consequence of the inequalities

$$f(\lambda(x)) \le u \quad \text{and} \quad |f(\lambda(x)) - f(\Theta(x))|_\infty \le d(\lambda(x),\Theta(x)) \le C\delta.$$

The first of these holds as $\lambda(x) \in \mathcal{R}_0^{u,\delta}$. The second holds by Property a, which implies that $d(\lambda(x),\Theta(x)) \le C\delta$, together with the assumption that $f\colon M \to \mathbb{R}^N$ is 1-Lipschitz.

Next, let us verify the shrinking transformation condition. We need to show that the diagram

$$\begin{array}{ccccc} \mathcal{R}^{\bullet_1,\bullet_2} & \xrightarrow{\lambda^{-1}} & \mathrm{sd}\mathcal{R}^{\bullet_1,\bullet_2} & \xrightarrow{\tau'} & \mathcal{R}^{S\bullet_1,\bullet_2+C\bullet_1} \\ & \searrow^{s} & & & \downarrow{\scriptstyle s} \\ & & & & \mathcal{R}^{\bullet_1,\bullet_2+C\bullet_1} \end{array} \tag{4}$$

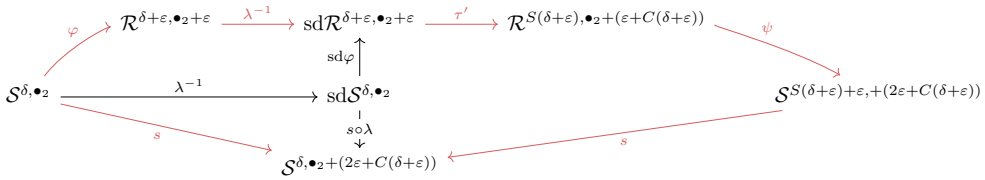
commutes. By precomposing with λ, this is equivalent to showing that the diagram

$$\begin{array}{ccc} \mathrm{sd}\mathcal{R}^{\bullet_1,\bullet_2} & \xrightarrow{\tau'} & \mathcal{R}^{S\bullet_1,\bullet_2+C\bullet_1} \\ {\scriptstyle\lambda}\downarrow & & \downarrow{\scriptstyle s} \\ \mathcal{R}^{\bullet_1,\bullet_2} & \xrightarrow[s]{} & \mathcal{R}^{\bullet_1,\bullet_2+C\bullet_1} \end{array} \tag{5}$$

commutes. In fact, it turns out that the two morphisms of persistent simplicial sets $s\circ\tau'$ and $s\circ\lambda$ are elementarily homotopic. To see this, consider the map

$$H\colon (\mathrm{sd}\mathcal{R}^{\bullet_1,\bullet_2}\times\Delta^1)_0 = (\mathrm{sd}\mathcal{R}^{\bullet_1,\bullet_2})_0 \times \{0,1\} \to M; \quad (x,i) \mapsto \begin{cases} \lambda(x), & i=0, \\ \Theta(x), & i=1, \end{cases}$$

and again apply Proposition 36 and Remark 38. As we already know that H_0 and H_1 fulfill the conditions of Proposition 36, it only remains to verify the condition on 1-simplices $\sigma \in (\mathrm{sd}\mathcal{R}^{\bullet_1,\bullet_2}\times\Delta^1)_1$ in Proposition 36 in the case where the vertices x and y are in $\mathrm{sd}\mathcal{R}^\delta(M)_0\times\{0\}$ and $\mathrm{sd}\mathcal{R}^\delta(M)_0\times\{1\}$ respectively. Observe that, by the definition of the simplicial product, we can identify σ with a simplex $\sigma' \in \mathrm{sd}\mathcal{R}^\delta(M)_1$, whose vertices we will also denote by x and y by abuse of notation. Then, the condition explicitly states that $d(\lambda(x),\Theta(y)) < \delta$ which follows from Property b in the definition of a pseudo-barycenter map. This shows that $s\circ\tau'$ and $s\circ\lambda$ are elementarily homotopic; thus, the shrinking transformation condition holds. It remains to show the compatibility with interleavings arising from a filtered correspondence. To this end, let $\mathfrak{C}$ be a filtered ε-correspondence between (M,f) and $(\mathbb{M},\mathbb{f})$, and let φ and ψ be the induced ε-interleaving maps. Furthermore, let $\varepsilon \le \min\frac{1-S}{1+S}\delta, \rho-\delta$. We write $\mathcal{S}^{\bullet_1,\bullet_2}$ for $\mathcal{R}^{\bullet_1}(\mathbb{M}^{\bullet_2})|_{[0,\rho]\times\mathbb{R}^N}$ and $\mathcal{S}^{\bullet_1}$ for $\mathcal{R}^{\bullet_1}(\mathbb{M})$. We now need to verify the commutativity of the outer diagram in



which we have marked in red. Observe that the left square commutes by the naturality of λ, and that the lower left triangle commutes by definition. It thus suffices to show that the remaining cell to the right commutes. To this end, we show that the two morphisms of persistent simplicial sets $s \circ \psi \circ \tau' \circ \mathrm{sd}\varphi$ and $s \circ \lambda$ are elementarily homotopic. To see this, we again use Proposition 36 and Remark 38, but this time applied to $\mathrm{sd}\mathcal{S}^{\delta,\bullet_2} \times \Delta^1$. Arguing exactly as above, we define a homotopy by extending the map

$$H' \colon (\mathrm{sd}\mathcal{S}^{\delta,\infty} \times \Delta^1)_0 = (\mathrm{sd}\mathcal{S}^{\delta})_0 \times \{0,1\} \to \mathbb{M}; \quad (x,i) \mapsto \begin{cases} \lambda(x), & i = 0. \\ (\psi \circ \tau' \circ \mathrm{sd}\varphi)(x), & i = 1; \end{cases}$$

By the same arguments as above, we can reduce to proving the following condition on 1-simplices $\sigma \in (\mathrm{sd}\mathcal{S}^{\delta,\bullet_2} \times \Delta^1)_1$ in Proposition 36 in the case where the vertices x and y are in $\mathrm{sd}(\mathcal{S}^{\delta,\bullet_2})_0 \times \{0\}$ and $\mathrm{sd}(\mathcal{S}^{\delta,\bullet_2})_0 \times \{1\}$, respectively: Given such a σ, it holds that

$$d(\lambda(x), (\psi \circ \tau' \circ \mathrm{sd}\varphi)(y)) < \delta.$$

To see this, we first apply the defining property of a correspondence to obtain

$$d(\lambda(x), (\psi \circ \tau' \circ \mathrm{sd}\varphi)(y)) \leq \varepsilon + d(\varphi(\lambda(x)), (\tau' \circ \mathrm{sd}\varphi)(y)) = \varepsilon + d(\varphi(\lambda(x)), \Theta((\mathrm{sd}\varphi)(y))).$$

By naturality of λ, we have $\varphi(\lambda(x)) = \lambda((\mathrm{sd}\varphi)(x))$. Hence, we only need to show that $d(\lambda(\mathrm{sd}\varphi(x)), \Theta((\mathrm{sd}\varphi)(y))) < \delta - \varepsilon$. By assumption, there exists a 1-simplex $\sigma' \in \mathrm{sd}\mathcal{S}_1^{\delta,\bullet_2}$ from x to y. Consequently, $\mathrm{sd}\varphi(\sigma') \in \mathrm{sd}\mathcal{R}_1^{\delta+\varepsilon,\bullet_2+\varepsilon}$ is a 1-simplex from $(\mathrm{sd}\varphi)(x)$ to $(\mathrm{sd}\varphi)(y)$ with $\delta + \varepsilon \leq \rho$. By Property b, we thus have

$$d(\lambda(\mathrm{sd}(\varphi)(x)), \Theta((\mathrm{sd}\varphi)(y))) < S(\delta + \varepsilon).$$

Finally, the assumption $\varepsilon \leq \frac{1-S}{1+S}\delta$ implies that $S(\delta+\varepsilon) \leq \delta - \varepsilon$, which concludes the proof. ◀

We can now finally combine the main results of this article to prove Theorem 3.

Proof of Theorem 3. By Proposition 34, the filtered correspondence $(\mathbb{M}, \mathbb{f}) \approx_\varepsilon (M, f)$ gives rise to an interleaving $\varphi \colon \mathcal{R}^\bullet(\mathbb{M}^\bullet) \simeq_\varepsilon \mathcal{R}^\bullet(M^\bullet) \colon \psi$ in the homotopy category. By Example 49, (M, f) admits a pseudo-barycenter map Θ with respect to $\rho = \rho_M^\kappa$ and $C = S = \frac{3}{4}$. Observe that $\frac{1-S}{1+S} = \frac{1}{7}$. By Theorem 51, Θ induces a shrinking transformation τ compatible with φ and ψ. Applying Theorem 44, we obtain an interleaving $\mathcal{R}^\delta(\mathbb{M}^\bullet) \simeq_{\frac{7}{4}\varepsilon + \frac{3}{4}\delta} \mathcal{R}^\delta(M^\bullet)$. Finally, since $\varepsilon \leq \frac{1}{7}\delta$ we have $\frac{7}{4}\varepsilon + \frac{3}{4}\delta \leq \delta$, and we obtain an interleaving $\mathcal{R}^\delta(\mathbb{M}^\bullet) \simeq_\delta \mathcal{R}^\delta(M^\bullet)$. ◀

References

1 Michał Adamaszek and Henry Adams. The Vietoris–Rips complexes of a circle. *Pacific Journal of Mathematics*, 290(1):1–40, 2017.

2 Michal Adamaszek, Henry Adams, and Florian Frick. Metric reconstruction via optimal transport. *SIAM Journal on Applied Algebra and Geometry*, 2(4):597–619, 2018. doi:10.1137/17M1148025.

3 Michał Adamaszek, Henry Adams, and Samadwara Reddy. On Vietoris–Rips complexes of ellipses. *Journal of Topology and Analysis*, 11(03):661–690, 2019.

4 Ethan André, Jingyi Li, David Loiseaux, and Steve Y. Oudot. Estimating the persistent homology of $\mathbb{r}^n$-valued functions using function-geometric multifiltrations. *arXiv preprint*, 2024. arXiv:2412.04162.

5 Omar Antolín-Camarena. Turning simplicial complexes into simplicial sets. unpublished notes, available at https://www.matem.unam.mx/~omar/notes/ssets-from-complexes.html. accessed 26 July 2021.

6 Clark Barwick and Daniel M. Kan. Relative categories: another model for the homotopy theory of homotopy theories. *Indagationes Mathematicae*, 23(1–2):42–68, 2012. doi:10.1016/j.indag.2011.10.002.

7 Julia E. Bergner. A survey of $(\infty, 1)$-categories. In John C. Baez and J. Peter May, editors, *Towards Higher Categories*, volume 152 of *IMA Volumes in Mathematics and its Applications*, pages 69–83. Springer, 2009. doi:10.1007/978-1-4419-1524-5_2.

8 Andrew J. Blumberg and Michael Lesnick. Stability of 2-parameter persistent homology. *Foundations of Computational Mathematics*, 24(2):385–427, 2024. doi:10.1007/S10208-022-09576-6.

9 Martin R. Bridson and André Haefliger. *Metric Spaces of Non-Positive Curvature*, volume 319 of *Grundlehren der Mathematischen Wissenschaften*. Springer, Berlin, 1999. doi:10.1007/978-3-662-12494-9.

10 Mickaël Buchet, Frédéric Chazal, Tamal K. Dey, Fengtao Fan, Steve Y. Oudot, and Yusu Wang. Topological analysis of scalar fields with outliers. In *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 827–841. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.SOCG.2015.827.

11 Gunnar Carlsson and Facundo Mémoli. Multiparameter hierarchical clustering methods. In Hermann Locarek-Junge and Claus Weihs, editors, *Classification as a Tool for Research*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 63–70. Springer, Berlin, Heidelberg, 2010. doi:10.1007/978-3-642-10745-0_6.

12 Frédéric Chazal, David Cohen-Steiner, Leonidas J. Guibas, Facundo Mémoli, and Steve Y. Oudot. Gromov-hausdorff stable signatures for shapes using persistence. *Computer Graphics Forum*, 28(5):1393–1403, 2009. doi:10.1111/j.1467-8659.2009.01516.x.

13 Frédéric Chazal, Vin de Silva, and Steve Y. Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 173(1):193–214, 2014. doi:10.1007/s10711-013-9937-z.

14 Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primož Skraba. Scalar field analysis over point cloud data. *Discrete & Computational Geometry*, 46(4):743–775, 2011. doi:10.1007/s00454-011-9360-x.

15 Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. Persistence-based clustering in Riemannian manifolds. *Journal of the ACM*, 60(6):41:1–41:38, 2013. doi:10.1145/2535927.

16 Denis-Charles Cisinski. *Higher Categories and Homotopical Algebra*. Number 180 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2019.

17 Vin de Silva, Elizabeth Munch, and Anastasios Stefanou. Theory of interleavings on categories with a flow. *Theory and Applications of Categories*, 33(21):583–607, 2018.

18 Oliver Gäfvert. *Topological and geometrical methods in data analysis*. PhD thesis, KTH Royal Institute of Technology, 2021.

19 Patrick Gillespie. Vietoris thickenings and complexes are weakly homotopy equivalent. *Journal of Applied and Computational Topology*, 8(1):35–53, 2024. doi:10.1007/s41468-023-00135-8.

20 Paul G. Goerss and John F. Jardine. *Simplicial Homotopy Theory*, volume 174 of *Progress in Mathematics*. Birkhäuser, Basel, reprint of the 1999 original edition, 2009. doi:10.1007/978-3-0346-0189-4.

21 John F. Jardine. Persistent homotopy theory. *arXiv preprint arXiv:2002.10013*, 2020. Revised version dated October 26, 2020. arXiv:2002.10013.

22 Daniel M. Kan. On c.s.s. complexes. *American Journal of Mathematics*, 79(3):449–476, 1957. doi:10.2307/2372450.

23 Rafal Komendarczyk, Sushovan Majhi, and Will Tran. Topological stability and Latschev-type reconstruction theorems for spaces of curvature bounded above. *arXiv preprint arXiv:2406.04259*, 2024. arXiv:2406.04259.

24 Edoardo Lanari and Luis Scoccola. Rectification of interleavings and a persistent Whitehead theorem. *Algebraic & Geometric Topology*, 23(2):803–832, 2023. doi:10.2140/agt.2023.23.803.

25 Janko Latschev. Vietoris–Rips complexes of metric spaces near a closed Riemannian manifold. *Archiv der Mathematik*, 77(6):522–528, 2001. doi:10.1007/PL00000526.

26 Boštjan Lemež and Žiga Virk. Finite reconstruction with selective Rips complexes. *arXiv preprint arXiv:2205.05525*, 2022. arXiv:2205.05525.

27 Sunhyuk Lim, Facundo Mémoli, and Osman Berat Okutan. Vietoris–Rips persistent homology, injective metric spaces, and the filling radius. *Algebraic & Geometric Topology*, 24(2):1019–1100, 2024. doi:10.2140/agt.2024.24.1019.

28 Jacob Lurie. *Higher Topos Theory*, volume 170 of *Annals of Mathematics Studies*. Princeton University Press, 2009. doi:10.1515/9781400830558.

29 Sushovan Majhi. Demystifying Latschev's theorem: Manifold reconstruction from noisy data. *Discrete & Computational Geometry*, 74(2):544–568, 2025. doi:10.1007/s00454-024-00655-9.

30 Nina Otter. Magnitude meets persistence. Homology theories for filtered simplicial sets. *Homology, Homotopy and Applications*, 24(2):365–387, 2022. doi:10.4310/HHA.2022.v24.n2.a12.

31 Steve Oudot and Lukas Waas. Function-Rips complexes in persistent homotopy theory: Stability and Latschev theorems. *arXiv preprint arXiv:2603.23460*, 2026. arXiv:2603.23460.

32 Daniel G. Quillen. *Homotopical Algebra*, volume 43 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin – New York, 1967. doi:10.1007/BFb0097438.

33 Frank Olaf Schreyer. A standard basis approach to syzygies of canonical curves. *Journal für die reine und angewandte Mathematik*, 421:83–124, 1991.

34 Matthew C. B. Zaremsky. Contractible Vietoris–Rips complexes of $\mathbb{z}^n$. *Proceedings of the American Mathematical Society*, 154(2):503–508, 2026.

Erdős's Unit Distance Problem and Rigidity

János Pach
Alfréd Rényi Institute of Mathematics, Budapest, Hungary

Orit E. Raz
Ben-Gurion University of the Negev, Beer Sheva, Israel

József Solymosi
University of British Columbia, Vancouver, Canada
Obuda University, Budapest, Hungary

Abstract

According to a classical result of Spencer, Szemerédi, and Trotter (1984), the maximum number of times the unit distance can occur among n points in the plane is $O(n^{4/3})$. This is far from Erdős's lower bound, $n^{1+O(1/\log\log n)}$, which is conjectured to be optimal. We prove a structural result for point sets with nearly $n^{4/3}$ unit distances and use it to reduce the problem to a conjecture on rigid frameworks. This conjecture, if true, would yield the first improvement on the bound of Spencer *et al.* A weaker version of this conjecture has been established by Raz and Solymosi.

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems

Keywords and phrases Unit distance problem, Erdős, graph rigidity, incidences, polynomial partitioning technique

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.83

Funding *János Pach*: This work was supported by ERC Advanced Grant no. 882971 "GeoScape" and by the National Research, Development and Innovation Office NKFIH Grant no. K-131529.
Orit E. Raz: Part of this research was supported by the Charles Simonyi Endowment.
József Solymosi: This research was supported by an NSERC Discovery grant and by the National Research Development and Innovation Office of Hungary, NKFIH, Grants no. KKP133819 and Excellence 151341.

Acknowledgements The authors thank Zvi Shem-Tov, Joshua Zahl, and Frank de Zeeuw for their valuable discussions and assistance in shaping the paper. Part of this research was conducted while the second author was a Member at the Institute for Advanced Study in Princeton.

1 Introduction

1.1 Background

In 1946, Paul Erdős [6, 7] raised two different problems for the distribution of distances among n points in the plane or in any fixed metric space S:

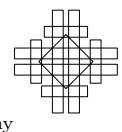
Problem A. What is the maximum number of times the same distance can occur among n points in S?

Problem B. What is the minimum number of distinct distances determined by n points of S?

The problems generated a lot of research and led to over a thousand papers. This is partially because they turned out to be related to many deep questions in number theory, combinatorics, Fourier analysis, algebraic, incidence, and computational geometry [8, 10, 26, 27]. For many related problems, see the monographs [2, 13, 16].


© János Pach, Orit E. Raz, and József Solymosi;
licensed under Creative Commons License CC-BY 4.0





The two problems are obviously interconnected. If we obtain an upper bound $u(n)$ for the quantity in Problem A, it gives the lower bound $\binom{n}{2}/u(n)$ for the number of distinct distances in Problem B. In particular, Erdős conjectured that in the plane $u(n) = n^{1+O(1/\log\log n)}$, which is attained by a $\sqrt{n} \times \sqrt{n}$ piece of the square grid. A matching upper bound is known only in exceptional cases where the vectors determining the unit distances are from a restricted family [20, 19].

Unfortunately, despite many efforts, the best known upper bound for the number of times that (say) the *unit distance* can occur among n points in the plane is

$$u(n) = O(n^{4/3}). \tag{1}$$

This was first proved by Spencer, Szemerédi, and Trotter [23]. Since then, the problem has been tackled by Clarkson, Edelsbrunner, Guibas, Sharir, and Welzl [4], Székely [24], Pach and Tardos [17], using different approaches, all yielding precisely the same upper bound, $O(n^{4/3})$. None of these approaches offered much hope for possible improvement.

To partially explain why all previous methods yielded the same upper bound, we reformulate the unit distance problem as an incidence problem between points and unit circles in $\mathbb{R}^2$: Let P be a set of n points in $\mathbb{R}^2$, and let C be a set of n unit circles in $\mathbb{R}^2$. Consider the set of point-circle incidences

$$I(P, C) = \{(p, c) \in P \times C \mid p \in c\}.$$

Observe that

$$\max_{|P|=n, |C|=n} I(P, C) = \Theta(u(n)).$$

That is, the problem of upper bounding $u(n)$ is equivalent to the problem of upper bounding the number of incidences between n points and n unit circles in $\mathbb{R}^2$.

Bounding the number of incidences between points and curves in $\mathbb{R}^d$ is a well-studied problem in combinatorial geometry. Specifically, the instance of point-line incidences in $\mathbb{R}^2$ is the classical result of Szemerédi and Trotter from 1984:

▶ **Theorem 1** (Szemerédi–Trotter [25]). *Let P be a set of m points in $\mathbb{R}^2$, and let L be a set of n lines in $\mathbb{R}^2$. Then*

$$I(P, L) = O(m^{2/3}n^{2/3} + m + n).$$

Moreover, this bound is tight.

In the special case $m = n$, Theorem 1 yields $I(P, L) = O(n^{4/3})$, and this bound is best possible. If we want to obtain a better bound for the incidence problem between points and unit circles in the plane, one has to come up with a method that does not apply to point-line incidences; otherwise, we are doomed to fail. This is "the" reason why the unit distance problem appears to be so difficult. All known methods, with the exception of the one developed in [17], also apply to the point-line incidence problem, hence, they cannot yield any bound better than $O(n^{4/3})$. The approach in [17] can be used to bound the number of incidences between points and some other families of curves, for which the bound is tight; see [28, 21].

As for Erdős's other problem (Problem B), concerning the smallest number of distinct distances, the lower bound has been steadily improved over the years by Moser [15], Chung, Szemerédi, and Trotter [3], Solymosi and Cs. Tóth [22], and Katz and Tardos [12]. In May

2010, Elekes and Sharir reduced the question to an incidence problem in $\mathbb{R}^3$. A couple of months later, in November of the same year, Guth and Katz [11] achieved a major breakthrough by solving the incidence problem of Elekes and Sharir. They deduced that any set of n points in the plane determines at least $cn/\log n$ distinct distances, where $c > 0$ is a suitable constant. This is only a factor of $\sqrt{\log n}$ smaller than the conjectured minimum, which is attained again by the square grid construction.

1.2 Graph rigidity

Before formulating our results, we need to recall some basic notions from graph rigidity theory.

We review some standard definitions from rigidity theory. For more details, see, e.g., Asimow and Roth [1]. Let $G = (V, E)$ be a graph. A *realization* $\mathbf{p}$ of G in $\mathbb{R}^2$ is an embedding (not necessarily injective) of the vertex set $V = \{1, \ldots, |V|\}$ of G in $\mathbb{R}^2$. That is,

$$\mathbf{p} = (p_1, .., p_{|V|}) \in \mathbb{R}^2 \times ... \times \mathbb{R}^2 \cong \mathbb{R}^{2|V|}.$$

A pair $(G, \mathbf{p})$ of a graph G and a realization $\mathbf{p}$ is called a *framework*.

Define the *edge function* of G to be the map $f_G : \mathbb{R}^{2|V|} \to \mathbb{R}^{|E|}$ given by

$$(p_1, \ldots, p_{|V|}) \mapsto \left(||p_i - p_j||^2\right)_{\{i,j\} \in E},$$

where we fix an arbitrary order on the edges of G. Note that f_G is a polynomial map.

For a given graph G, let $\mathbf{p}$ and $\mathbf{q}$ be a pair of realizations of G in $\mathbb{R}^2$. We say that the corresponding frameworks, $(G, \mathbf{p})$ and $(G, \mathbf{q})$, are *equivalent* if $f_G(\mathbf{p}) = f_G(\mathbf{q})$, that is, if $\|p_i - p_j\| = \|q_i - q_j\|$ holds for every edge $\{i, j\} \in E$. We say that the frameworks $(G, \mathbf{p})$ and $(G, \mathbf{q})$ are *congruent* if $\|p_i - p_j\| = \|q_i - q_j\|$ for every size-2 subset $\{i, j\} \subset V$ (here $\{i, j\}$ is not necessarily an edge in G). Equivalently, $(G, \mathbf{p})$ and $(G, \mathbf{q})$ are congruent if there exists an isometry R of $\mathbb{R}^2$ such that $R(p_i) = q_i$, for every $i \in V$.

▶ **Definition 2** (Framework Rigidity). *We say that* $(G, \mathbf{p})$ *is a* rigid framework *if there exists a neighborhood* $U \subset (\mathbb{R}^2)^{|V|}$ *of* $\mathbf{p}$*, such that for every* $\mathbf{q} \in U$*, if* $(G, \mathbf{p})$ *and* $(G, \mathbf{q})$ *are equivalent, then they are necessarily also congruent. Equivalently, if there exists a neighborhood* U *of* $\mathbf{p}$ *such that*

$$f_G^{-1}(f_G(\mathbf{p})) \cap U = f_{K_{|V|}}^{-1}(f_{K_{|V|}}(\mathbf{p})) \cap U,$$

where $K_{|V|}$ *is the complete graph on* $|V|$ *vertices.*

Note that the notion of framework rigidity depends on both the graph G and the realization $\mathbf{p}$. That is, for a given graph G, and realizations $\mathbf{p}$ and $\mathbf{q}$, it might happen that $(G, \mathbf{p})$ is rigid and $(G, \mathbf{q})$ is not rigid.

A realization of G is called *generic* if the set of all coordinates of all of its points is algebraically independent over the rationals. It turns out that all *generic* realizations of G behave the same. That is, if G is fixed and $\mathbf{p}$ and $\mathbf{q}$ are some generic realizations of G, then $(G, \mathbf{p})$ is rigid if and only if $(G, \mathbf{q})$ is rigid. In this sense, rigidity is a property of the graph G, independent of the specific generic realization $\mathbf{p}$ one considers.

▶ **Definition 3** (Graph Rigidity). *We say that* $G = ([n], E)$ *is a* rigid graph *in* $\mathbb{R}^2$*, if* $(G, \mathbf{p})$ *is a rigid framework, for every generic realization* $\mathbf{p} \in (\mathbb{R}^2)^n$.

Most results in rigidity theory are concerned with the notion of generic rigidity, as in Definition 3. In the present note, we study realizations $\mathbf{p}$ arising from configurations of n points in the plane that maximize the number of unit distances. Such configurations are highly *non-generic*. Hence, for us *framework rigidity*, as in Definition 2, will be more relevant, and we will focus on this notion.

Specifically, given a framework $(G, \mathbf{p})$, we aim to find a rigid sub-framework within it. Note that this is an easy task if we restrict our attention to *generic* realizations. It is not hard to show that every sufficiently dense graph contains a *generically* rigid subgraph (see, e.g., [5] for technical details).

▶ **Lemma 4.** *Let $G = (V, E)$, let $\varepsilon > 0$ be fixed, and suppose that $|E| \geq n^{1+\varepsilon}$. Then there exists $G' \subset G$ with $|V(G')| \geq 3$ such that G' is generically rigid in $\mathbb{R}^2$.*

However, if our graph is given together with a realization, there is not much theory about this problem. The only useful result in this direction was obtained by Raz and Solymosi in [18].

▶ **Theorem 5** (Raz and Solymosi [18])**.** *Let $G = ([n], E)$, let $\alpha > 1/2$, and suppose that*

$$|E| = \Omega(n^{1+\alpha}). \tag{2}$$

Let $\mathbf{p} \in (\mathbb{R}^2)^n$ be a realization of G with the property that for every vertex $v \in [n]$, the neighbors of v are not embedded into a common line. Then there exists a subgraph $G' \subset G$, with $|V(G')| \geq 4$, such that $(G', \mathbf{p}|_{V(G')})$ is a rigid framework, provided n is large enough.

The smallest value of α for which the conclusion of Theorem 5 holds is not known. As was observed in [18], we need to assume at least $|E| = \Omega(n \log n)$; otherwise, the statement is false.

In [18], Theorem 5 was proved under the slightly stronger assumption that there are no *three* vertices of G that are mapped by $\mathbf{p} \in (\mathbb{R}^2)^n$ into collinear points. However, essentially the same proof also gives Theorem 5.

1.3 Main results

Our goal is to suggest a new approach to tackle the unit distance problem (Problem A) by reducing it to a rigidity problem.

For any point set P in the plane, let $u(P)$ denote the number of unit distance pairs determined by P, that is, the number of unordered pairs $\{p, q\} \subseteq P$ with $\|p - q\| = 1$. Given a graph $G = (V, E)$ and a realzation $\mathbf{p}$ of its vertex set into $\mathbb{R}^{2|V|}$ such that, $\mathbf{p}$ is injective, and the endpoints of every edge are mapped into two points whose distance is 1, we call $\mathbf{p}$ a *unit embedding* of G.

In the sequel, when we compare two functions, we will often use the notation $f(n) \lesssim g(n)$ instead of $f(n) = O(g(n))$. If we have $f(n) \lesssim g(n)$ and $g(n) \lesssim f(n)$, then we write $g(n) \approx h(n)$.

We tacitly assume that there exist n-element point sets P with $u(P)$ close to the currently best known upper bound $n^{4/3}$, and we study their structure. Our main technical result is the following.

▶ **Theorem 6** (Structure Theorem)**.** *Let $h(n)$ be a function tending to ∞, as $n \to \infty$. Let P be a set of n points in $\mathbb{R}^2$ satisfying $u(P) \geq \frac{n^{4/3}}{h(n)}$, and suppose that n is sufficiently large.*

Then there exist

1. *a subset $P' \subset P$ with $|P'| \approx n^{1/3} h(n)^4$;*
2. *bipartite graphs $G_i = (U_i \cup V_i, E_i)$ for every i $(1 \leq i \leq k)$, where $k \gtrsim n^{2/3}/h(n)^5$, such that $2 \leq |U_i|, |V_i| \lesssim h(n)^6$ and $|E_i| \gtrsim h(n)^7$;*

3. *unit embeddings* $\mathbf{p}^{(i)}$ *of* G_i *into* $(\mathbb{R}^2)^{|U_i|+|V_i|}$ *for every* i $(1 \leq i \leq k)$*, such that*
   - **(i)** *the sets* $\mathbf{p}^{(1)}(U_1), \ldots, \mathbf{p}^{(k)}(U_k)$ *are pairwise disjoint subsets of* P*,*
   - **(ii)** *the sets* $\mathbf{p}^{(1)}(V_1), \ldots, \mathbf{p}^{(k)}(V_k)$ *are subsets of* P'*.*

We conjecture that the statement of Theorem 5 remains true for any $\alpha \geq 1/6$. In other words, we state the following conjecture.

▶ **Conjecture 7** (Rigidity Conjecture)**.** *Let* $G = ([n], E)$ *be a graph with* $|E| \gtrsim n^{7/6}$*. Let* $\mathbf{p} \in (\mathbb{R}^2)^n$ *be a realization of* G *with the property that for every vertex* $v \in [n]$*, the neighbors of* v *are not embedded into a common line. Then there exists a subgraph* $G' \subset G$ *with* $|V(G')| \geq 4$ *such that* $(G', \mathbf{p}|_{V(G')})$ *is a rigid framework.*

Provided that Conjecture 7 is true, our approach would yield the first improvement of the classical upper bound, $O(n^{4/3})$, of Spencer, Szemerédi, and Trotter [23] on the number of unit distances for more than 40 years. Notably, we will deduce the following result.

▶ **Theorem 8.** *Conjecture 7 implies that the maximum number of unit distance pairs,* $u(n)$*, determined by* n *points in the plane satisfies*

$$u(n) = O\left(\frac{n^{4/3}}{\log^{1/12} n}\right).$$

The proofs of Theorems 8 and 6 are given in Sections 2 and 3, respectively.

2 Proof of Theorem 8

Let $P \subset \mathbb{R}^2$ with $|P| = n$, and assume for contradiction that $u(P) \geq \frac{n^{4/3}}{h(n)}$, for some function $h = h(n)$ that tends to zero as n tends to infinity. Let $P' \subset P$, k, $G_1 = (U_1 \cup V_1, E_1), \ldots, G_k = (U_k \cup V_k, E_k)$ and $\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(k)}$ be as given by Theorem 6.

Note that, since the realizations $\mathbf{p}^{(i)}$ are unit embeddings, the assumption in Conjecture 7, that the neighbors of a vertex v are not embedded to a common line, applies automatically. Thus, assuming that Conjecture 7 is true, it follows that, for every $i = 1, \ldots, k$, there exists a subgraph $G_i' = (U_i' \cup V_i', E_i') \subset G_i$ such that G_i' has at least 4 vertices and the unit framework $(G_i', \mathbf{p}^{(i)}|_{U_i' \cup V_i'})$ is rigid. Since G_i' is bipartite, this implies, in particular, that $|U_i|, |V_i| \geq 2$.

Recall that, by Theorem 6 item 2, we have that for each i, $2 \leq |U_i|, |V_i| \leq h^6$. Thus also $2 \leq |U_i'|, |V_i'| \leq h^6$. By the pigeonhole principle, there exists some bipartite graph $H = (U \cup V, E)$, with $2 \leq |U|, |V| \leq h^6$, such that for at least

$$k/2^{h^{12}},$$

indices i, we have that G_i' is isomorphic to H. Note that a rigid framework in the plane, on m vertices, has at most 9^m distinct non-congruent embeddings that induce the same edge lengths (see Milnor [14, Theorem 2]). Applying the pigeonhole principle once again, we conclude that there are at least

$$k/(2^{h^{12}} 9^{2h^6})$$

indices i, for which the frameworks $(G_i', \mathbf{p}^{(i)}|_{U_i' \cup V_i'})$ are pairwise congruent. Let I denote the subset of such indices. So $|I| \geq k/(2^{h^{12}} 9^{2h^6})$.

Let $v_1, v_2 \in V$. Then there exists a number $a > 0$ such that, for every $i \in I$, the embedding $\mathbf{p}^{(i)}$ maps v_1, v_2 to a pair of points in P' that are at distance a from each other.

We claim that for every $p, q \in P'$, there are at most two indices $i \in I$ such that v_1 is embedded by $\mathbf{p}^{(i)}$ to p and v_2 is embedded by $\mathbf{p}^{(i)}$ to q. Indeed, note that by fixing the embedding of v_1, v_2, we have that $\mathbf{p}^{(i)}$ must be one of at most two possible realizations of H (which can be obtained from each other by a reflection through the line pq). Thus, the embedding of U_i' is determined up to two possibilities. Since the sets $\mathbf{p}^{(i)}(U_i)$ are pairwise disjoint, it follows that i must be one of at most two possible indices in I. Similarly, there are at most two indices $i \in I$ such that $\mathbf{p}^{(i)}$ embeds v_1 to q and v_2 to p.

We conclude that the number of pairs in P' determining distance a is at least

$$k/(4 \cdot 2^{h^{12}} \cdot 9^{2h^6}).$$

Since the same distance can be repeated at most $\lesssim |P'|^{4/3}$ times in $|P'|$, we have

$$k/(4 \cdot 2^{h^{12}} \cdot 9^{2h^6}) \lesssim |P'|^{4/3} \approx (n^{1/3}h^4)^{4/3}.$$

Using the lower bound on k, we obtain that

$$\frac{n^{2/3}}{h^5 2^{h^{12}+2h^6+1}} \lesssim n^{4/9}h^{16/3}$$

or

$$n^{2/9} \lesssim 2^{h^{12}+2\log(9)h^6+1+(31/3)\log h}.$$

This yields a contradiction if $h(n) \approx \log^{1/12} n$, completing the proof of the theorem. ◀

3 Proof of Theorem 6

For the proof, we need the following result of Guth [9].

▶ **Theorem 9** (Guth [9, Theorem 0.3]). *Let Γ be a set of k-dimensional varieties in $\mathbb{R}^d$, each defined by at most b polynomial equations of degree at most δ. Then, for any $D \geq 1$, there is a nonzero polynomial f of degree at most D such that each connected component of $\mathbb{R}^d \setminus \{f = 0\}$ intersects $\lesssim D^{k-d}|\Gamma|$ varieties $\gamma \in \Gamma$, where the constant of proportionality depends on d, δ, and b.*

Let $P \subset \mathbb{R}^2$ with $|P| = n$, where n is sufficiently large. Assume that

$$u(P) \geq \frac{n^{4/3}}{h(n)},$$

for some function $h = h(n)$, which tends to ∞ as $n \to \infty$.

Let C denote the set of unit circles centred at the points of P. By our assumption on P, we have that the number of incidences between P and C is

$$|I(P, C)| \geq \frac{n^{4/3}}{h}$$

First partitioning

Set

$$r = \frac{n^{1/3}}{h^2}.$$

By Theorem 9, there exists $f \in \mathbb{R}[x,y]$, with $\deg(f) \leq r$, whose zero set partitions the plane into $\lesssim r^2$ cells such that each cell contains at most $\lesssim n/r^2$ points of P and meets at most $\lesssim n/r$ unit circles of C.

The number of incidences that occur on the zero set of f is at most

$$|I_0(P,C)| \leq nr + n + r^2 < \frac{1}{2}|I(P,C)|,$$

if $n \geq n_0$ is sufficiently large.

Thus, in what follows, we assume without loss of generality that $P \subset \mathbb{R}^2 \setminus Z(f)$ and that no circle in C is contained in $Z(f)$.

Let Ω denote the set of open connected components of $\mathbb{R}^2 \setminus Z(f)$. For $\omega \in \Omega$, let P_ω denote the set of points in P contained in ω, and let C_ω denote the set of circles in C that intersect ω. By our choice of f, we have

$$|\Omega| \lesssim \frac{n^{2/3}}{h^4},$$

and, for every $\omega \in \Omega$,

$$|P_\omega| \lesssim n/r^2 = n^{1/3}h^4$$

and

$$|C_\omega| \lesssim n/r = n^{2/3}h^2.$$

Recall our assumption that all incidences in $I(P,C)$ occur within the cells. Then, by the pigeonhole principle, there exists a cell $\omega_0 \in \Omega$ such that

$$|I(P_{\omega_0}, C_{\omega_0})| \gtrsim \frac{n^{4/3}}{h} / \frac{n^{2/3}}{h^4} = n^{2/3}h^3.$$

Let Q denote the set of centers of the unit circles in C_{ω_0}. Let D denote the set of unit circles centred at the points of P_{ω_0}. We have

$$\begin{aligned} |Q| &\lesssim n^{2/3}h^2, \\ |D| &\lesssim n^{1/3}h^4, \quad \text{and} \\ |I(Q,D)| &\gtrsim n^{2/3}h^3. \end{aligned}$$

Second partitioning

Set

$$t \approx \frac{n^{1/3}}{h^2}.$$

Let $g \in \mathbb{R}[x,y]$ be a bivariate polynomial of degree at most t such that $\mathbb{R}^2 \setminus Z(g)$ partitions $\mathbb{R}^2$ into at most $\lesssim t^2$ connected components, each of which contains at most $\lesssim |Q|/t^2 \lesssim h^6$ points of Q, and meets at most $\lesssim |D|/t \lesssim h^6$ unit circles in D.

Note that the number of incidences in $I(Q,D)$ that occur on the zero set of g is bounded by

$$|I_0(Q,D)| \leq t|D| + |Q| + t^2 < \frac{1}{2}|I(Q,D)|,$$

for $n \geq n_0$ sufficiently large. Thus, without loss of generality, we may assume that all incidences in $I(Q,D)$ occur within the cells.

Let Π denote the open connected components of $\mathbb{R}^2 \setminus Z(g)$. For $\pi \in \Pi$, let Q_π denote the set of points of Q contained in π, and let D_π denote the set of circles in D that meet π. By our choice of g, we get

$$|\Pi| \lesssim \frac{n^{2/3}}{h^4}$$

and, for every $\pi \in \Pi$, we have

$$|Q_\pi| \lesssim h^6 \quad \textit{and}$$
$$|D_\pi| \lesssim h^6,$$

as was already noted above.

Note that cells π with at most $\leq ch^7$ incidences, for some constant $c > 0$, contribute a total of at most

$$\lesssim \frac{n^{2/3}}{h^4} \cdot ch^7 = cn^{2/3}h^3 < \frac{1}{2}I(Q, D)$$

incidences, provided that $c > 0$ is chosen sufficiently small.

Let $\Pi' \subset \Pi$ be the subset of cells π for which $|I(Q_\pi, D_\pi)| \geq ch^7$. Observe that

$$|\Pi'| \gtrsim \frac{n^{2/3}}{h^5}. \tag{3}$$

Indeed, by the Szemerédi–Trotter bound, in each cell $\pi \in \Pi$, we have $|I(Q_\pi, D_\pi)| \lesssim (|Q_\pi||D_\pi|)^{2/3} \approx h^8$. Thus, we have

$$|\Pi'| \cdot h^8 \gtrsim \frac{1}{2}|I(Q, D)| \gtrsim n^{2/3}h^3,$$

which implies (3).

Finally, let $P' := P_{\omega_0}$, or, equivalently, the centers of the circles in D. Note that $P' \subset P$. For every $\pi \in \Pi'$, let G_π denote the incidence graph between the points in Q_π and the unit circles in D_π. Observe that G_π is a bipartite graph with parts Q_π, D_π, each of cardinality at most $\lesssim h^6$, and that the number of edges in G_π is $|E(G_\pi)| \gtrsim h^7$. Moreover, identifying the circles in D_π with their centers, we get that (Q_π, D_π) is a *unit embedding* of G_π, by construction. Observing also that the sets $Q_\pi \subset P$ are pairwise disjoint (by the properties of the partition induced by the partitioning polynomial g), and in view of (3), the proof of Theorem 6 is complete. ◀

References

1 Leonard Asimow and Ben Roth. The rigidity of graphs. *Trans. Amer. Math. Soc.*, 245:279–289, 1978.

2 Peter Brass, William O. J. Moser, and János Pach. *Research Problems in Discrete Geometry*. Springer, New York, 2005.

3 Fan Chung, Endre Szemerédi, and William T. Trotter. The number of different distances determined by a set of points in the euclidean plane. *Discrete & Computational Geometry*, 7(1):1–11, 1992. doi:10.1007/BF02187820.

4 Kenneth L. Clarkson, Herbert Edelsbrunner, Leonidas J. Guibas, Micha Sharir, and Emo Welzl. Combinatorial complexity bounds for arrangements of curves and spheres. *Discrete & Computational Geometry*, 5(2):99–160, 1990. doi:10.1007/BF02187783.

5 Robert Connelly. Generic global rigidity. *Discrete & Computational Geometry*, 33:549–563, 2005. doi:10.1007/S00454-004-1124-4.

6 Paul Erdős. On sets of distances of n points. *American Mathematical Monthly*, 53(5):248–250, 1946.
7 Paul Erdős. On sets of distances of n points in euclidean space. *Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei*, 5(1–2):165–169, 1959.
8 Julia Garibaldi, Alex Iosevich, and Steven Senger. *The Erdős Distance Problem*. American Mathematical Society, 2011.
9 Larry Guth. Polynomial partitioning for a set of varieties. *Mathematical Proceedings of the Cambridge Philosophical Society*, 159:459–469, 2015.
10 Larry Guth. *Polynomial Methods in Combinatorics*. American Mathematical Society, 2016.
11 Larry Guth and Nets Hawk Katz. On the erdős distinct distances problem in the plane. *Annals of Mathematics*, pages 155–190, 2015.
12 Nets Hawk Katz and Gábor Tardos. A new entropy inequality for the erdős distance problem. *Contemporary Mathematics*, 342:119–126, 2004.
13 Jiři Matoušek. *Lectures on Discrete Geometry*. Springer Science & Business Media, 2013.
14 John Milnor. On the betti numbers of real varieties. *Proceedings of the American Mathematical Society*, 15:275–280, 1964.
15 Leo Moser. On the different distances determined by n points. *American Mathematical Monthly*, 59(2):85–91, 1952.
16 János Pach and Pankaj K. Agarwal. *Combinatorial Geometry*. John Wiley & Sons, 2011.
17 János Pach and Gábor Tardos. Forbidden paths and cycles in ordered graphs and matrices. *Israel Journal of Mathematics*, 155(1):359–380, 2006.
18 Orit E. Raz and József Solymosi. Dense graphs have rigid parts. *Discrete & Computational Geometry*, 69:1079–1094, 2023. doi:10.1007/S00454-022-00477-7.
19 Ryan Schwartz. Using the subspace theorem to bound unit distances. *Moscow Journal of Combinatorics and Number Theory*, 3(1):108–117, 2013.
20 Ryan Schwartz, József Solymosi, and Frank de Zeeuw. Rational distances with rational angles. *Mathematika*, 58(2):409–418, 2012.
21 József Solymosi and Endre Szabó. Classification of maps sending lines into translates of a curve. *Linear Algebra and its Applications*, 668:161–172, 2023.
22 József Solymosi and Csaba D. Tóth. Distinct distances in the plane. *Discrete & Computational Geometry*, 25:629–634, 2001. doi:10.1007/S00454-001-0009-Z.
23 Joel Spencer, Endre Szemerédi, and William T. Trotter. Unit distances in the euclidean plane. In *Graph Theory and Combinatorics*, pages 294–304. Academic Press, 1984.
24 László Székely. Crossing numbers and hard erdős problems in discrete geometry. *Combinatorics, Probability and Computing*, 6(3):353–358, 1997. URL: http://journals.cambridge.org/action/displayAbstract?aid=46513.
25 Endre Szemerédi and William T. Trotter. Extremal problems in discrete geometry. *Combinatorica*, 3:381–392, 1983. doi:10.1007/BF02579194.
26 Terence Tao. Algebraic combinatorial geometry: the polynomial method in arithmetic combinatorics, incidence combinatorics, and number theory. *EMS Surveys in Mathematical Sciences*, 1(1):1–46, 2014.
27 Terence Tao and Van H. Vu. *Additive Combinatorics*. Cambridge University Press, 2006.
28 P. Valtr. Strictly convex norms allowing many unit distances and related touching questions, 2005. Manuscript.

The Voronoi Diagram of Four Lines in $\mathbb{R}^3$

Evanthia Papadopoulou
Faculty of Informatics, Università della Svizzera italiana (USI), Lugano, Switzerland

Zeyu Wang
Faculty of Informatics, Università della Svizzera italiana (USI), Lugano, Switzerland

Abstract

We consider the Voronoi diagram of lines in $\mathbb{R}^3$ under the Euclidean metric, and give a full classification of its structure in the base case of four lines in general position. We first show that the number of vertices in the Voronoi diagram of four lines in general position is always even, between 0 and 8, and all such numbers can be realized. We identify a key structure for the diagram formation, called a *twist*, which is a pair of consecutive intersections among trisector branches; only two types of twists are possible, so-called *full* and *partial* twists. A full twist is a purely local structure, which can be inserted or removed without affecting the rest of the diagram. Assuming no full twists, the nearest and the farthest Voronoi diagrams of four lines, each have 15 distinct topologies, which are in one-to-one correspondence; the two-dimensional faces are all unbounded, and the total number of vertices is at most six. The unbounded features of the farthest diagram, encoded in a two-dimensional spherical map, are also in one-to-one correspondence. The identified topologies are all realizable. Any Voronoi diagram of four lines in general position in $\mathbb{R}^3$ can be obtained from one of these topologies by inserting full twists; each twist induces a bounded face of exactly two vertices in both the nearest and farthest diagrams. We obtain the classification by an exhaustive search algorithm using some new structural and combinatorial observations of line Voronoi diagrams.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Voronoi diagram, lines, three dimensions, structural properties

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.84

Related Version *Full Version*: `https://arxiv.org/abs/2603.19836`

Supplementary Material *Software*: `https://doi.org/10.5281/zenodo.19090548` [22]

Funding Supported by the Swiss National Science Foundation (SNF), project 200021E_201356.

Acknowledgements We thank Dr. Martin Suderland for early discussions and valuable comments related to Sections 2, 3, and for two MATLAB-based visualisation tools, which allowed us to visualize $\Gamma(\mathrm{FVD}(L))$, Figure 6, and Figure 13. We also thank anonymous reviewers for comments that helped improve the presentation of this paper, the general position assumption, and Lemma 3.

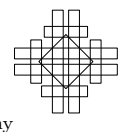
1 Introduction

Voronoi diagrams are fundamental space partitioning structures in Computational Geometry. Given a set S of n objects in some space, called sites, the *nearest* (respectively, *farthest*) Voronoi diagram of S decomposes the underlying space into regions that have the same nearest (resp., farthest) site. In this paper, we consider Voronoi diagrams of lines in $\mathbb{R}^3$ under the Euclidean metric, focusing on the elementary, yet fundamental base case of four lines.

The Euclidean Voronoi diagram of point sites in $\mathbb{R}^d$ is a classical geometric partitioning structure that is generally well-understood. It has complexity $O(n^{\lceil \frac{d}{2} \rceil})$ and can be computed in $O(n \log n + n^{\lceil \frac{d}{2} \rceil})$ time [7, 10, 16]; these bounds are tight. The results also hold for certain polyhedral norms such as the L_∞ or L_1 norms [6, 15]. For the Euclidean farthest Voronoi diagram of point sites, exact worst-case bounds in the range of $\Theta(n^{\lceil \frac{d}{2} \rceil})$ have also been reported by Seidel [20].


© Evanthia Papadopoulou and Zeyu Wang;
licensed under Creative Commons License CC-BY 4.0





For sites more general than points, however, Voronoi diagrams in $\mathbb{R}^d$, $d \geq 3$, have not yet been well-understood. These diagrams get complicated because their bisecting surfaces are curved, triggering complicated algebraic descriptions of their features such as their trisector curves. The combinatorial complexity of the classical Voronoi diagram of n lines (or line segments) in $\mathbb{R}^3$ is a well-known outstanding open problem [19]. There is a gap of an order of magnitude between the $\Omega(n^2)$ lower bound [2] and the only known $O(n^{3+\varepsilon})$ upper bound by Sharir [21]. This gap extends to the Voronoi diagram of lines in $\mathbb{R}^d$, $d > 3$, where the known upper bound is $O(n^{d+\varepsilon})$ [21]. For $d \geq 4$, an $\Omega(n^{\lceil 2d/3 \rceil})$ lower bound has recently been reported by Glisse [13]. The upper bounds are derived from general complexity results on envelopes of algebraic functions in $\mathbb{R}^{d+1}$, under the general framework [10] for studying Voronoi diagrams in $\mathbb{R}^d$ through the *arrangement* of distance functions of the given sites in $\mathbb{R}^{d+1}$. Sharper complexity bounds have been established in certain special cases such as lines with c constant orientations [18] and parallel half-lines [3], and for norms induced by convex polyhedra of constant complexity [4, 8, 17].

The Voronoi diagram of only three lines in $\mathbb{R}^3$ was addressed by the authors of [12] who proved the fundamental properties of its structure. The trisector of three pairwise skew lines was shown to consist of four unbounded curves, which are roots of a nonsingular quartic, or a nonsingular cubic and a nonintersecting line. This is the first, most basic building block necessary in computing any Voronoi diagram of lines or polyhedra in $\mathbb{R}^3$, but it contains no Voronoi vertices. In this paper, we address the next basic building block, which is the Voronoi diagram of four lines in $\mathbb{R}^3$, where a minimal number of Voronoi vertices appears.

The unbounded features of the Euclidean order-k, $1 \leq k \leq n-1$, Voronoi diagrams of lines and line segments in $\mathbb{R}^d, d \geq 3$, were studied in [5]. They are encoded in a map on a *sphere of directions*, called the Gaussian map, which has complexity $O(\min\{k, n-k\}n^{d-1})$; for the farthest Voronoi diagram the Gaussian map has complexity $\Theta(n^{d-1})$.

The Voronoi diagram of lines in $\mathbb{R}^3$ can be computed in $O(n^{3+\epsilon})$ time [1] as part of a more general technique to compute the lower envelope of algebraic functions in $\mathbb{R}^4$. Using the envelope package of CGAL, a numerically robust algorithm for computing the Voronoi diagram of lines in $\mathbb{R}^3$ has been reported in [14]. Algorithms for computing approximations of the Voronoi diagram also exist, see e.g., [9, 11].

Contribution. In this paper, we prove structural properties of the Voronoi diagram of four lines in $\mathbb{R}^3$, building upon the previous work [12] on three lines. The case of four lines is fundamental, and is required for understanding any Voronoi diagram of lines or polyhedra in $\mathbb{R}^3$. We first show that the number of vertices in the Voronoi diagram of four lines in general position is always even, between 0 and 8, and all such numbers can be realized. We then identify basic structures in the arrangement of trisectors, called *twists*, which are critical for the formation of the diagram and the resulting number of its Voronoi vertices. A twist is a pair of consecutive trisector intersections along some trisector components (called branches, see Definition 10). We show that there are only two types of twists: *full twists* that involve four different trisector branches, and *partial twists* that involve six trisector branches, two of which are incident to both vertices of the twist (see Definition 11). A full twist has a *local* structure, in the sense that adding or removing one does not affect the rest of the diagram. It results in a bounded face with exactly two vertices in both the nearest and the farthest diagrams. Since full twists only have a local effect, we first assume that no full twists exist in the trisector arrangement and fully classify the Voronoi diagrams of four lines in this case. We prove the following theorems.

▶ **Theorem 1.** *The Voronoi diagram of four lines in general position in $\mathbb{R}^3$ (both the nearest and farthest diagram) has 15 distinct topologies, assuming there are no full twists among the trisectors. The distinct topologies of the nearest and the farthest diagrams are in one-to-one correspondence. Further, they are in one-to-one correspondence with the unbounded features of the farthest diagram. To obtain more than two Voronoi vertices, twists are required. The two-dimensional faces in both diagrams are unbounded and the total number of vertices is at most 6, see Table 1. In the nearest Voronoi diagram, all vertices are connected in a single component of the 1-skeleton.*

Table 1 Distinct topologies for each possible number of Voronoi vertices, assuming no full twists.

Number of vertices	0	2	4	6	8
Number of topologies	1	3	5	6	0

Additional structural properties are listed in Observation 21.

▶ **Theorem 2.** *The nearest (resp. farthest) Voronoi diagram of four lines in general position in $\mathbb{R}^3$ can be obtained from one of the 15 topologies of Theorem 1 by inserting the diagram's full twists. Inserting a full twist to a topology creates one new bounded face with two vertices in both the nearest and farthest diagram; it also splits an existing face in two faces. The total number of vertices, including full-twists, is at most 8.*

To obtain this classification, we devise an exhaustive search algorithm that enumerates possible *configurations* of projected trisector pairs. By ignoring the geometry and focusing on the structure, we encode a bisector and the two trisectors it contains as a *configuration* of eight curves in the plane, where each trisector has four branches. We identify necessary conditions under which a configuration is valid, and use them to create filters that eliminate impossible cases. We also create filters for six-tuples of configurations, where each tuple corresponds to the six bisectors of four lines. After an exhaustive search, which eliminates impossible cases by the derived necessary conditions, we are left with only 15 distinct configuration six-tuples. Each tuple corresponds to a distinct topology. The enumeration process is shown to be complete. All the remaining 15 tuples and their corresponding topologies are realizable as we demonstrate by concrete examples. Based on these results, we also provide an algebraically simple method to identify the structure of the Voronoi diagram.

In addition, we establish some simple new combinatorial results on the nearest and farthest Voronoi diagrams of n lines, expressing their features as a function of the number of Voronoi vertices (see Theorem 9). We use these results for $n = 4$ to obtain the aforementioned filters. For the farthest Voronoi diagram of n lines, we show that the region of each line has exactly $n-1$ three-dimensional cells, extending a result of [5] on the total number of cells.

The exhaustive search has been implemented as a program, available on Zenodo [22], that performs the actual search.

2 Preliminaries

Let $L = \{\ell_1, \ldots, \ell_n\}$ be a set of n lines in general position in $\mathbb{R}^3$. By general position, we mean the following: (1) lines are pairwise skew, and no three lines are parallel to a common plane; (2) no sphere can be tangent to four lines with coplanar tangency points; (3) no sphere is tangent to five lines; (4) no four lines have direction vectors whose representations on the sphere of directions are cocircular.

We denote by $d(x, y)$ the Euclidean distance between two points $x, y \in \mathbb{R}^3$. The distance $d(x, \ell)$ from a point $x \in \mathbb{R}^3$ to a line $\ell \in L$ is defined as $d(x, \ell) = \min\{d(x, y) \mid y \in \ell\}$. The i-sector of i lines in $\mathbb{R}^3$, $i \in \{2, 3\}$, is the locus of points equidistant from the i lines. For $i = 2$, a bisector $B(\cdot, \cdot)$ is a hyperbolic paraboloid; for $i = 3$, a trisector $T(\cdot, \cdot, \cdot)$ is a quartic consisting of four unbounded branches [12, 18]. Two trisectors are said to be *related* if they involve exactly four lines, i.e., they are of the form $T(\ell_i, \ell_j, \ell_k), T(\ell_i, \ell_j, \ell_l)$, with $k \neq l$.

The *nearest Voronoi diagram* of L, denoted $\mathrm{NVD}(L)$, subdivides $\mathbb{R}^3$ into maximal regions, each consisting of points closer to one line than to any other line. More generally, for $1 \leq k \leq n-1$, the *order-k Voronoi diagram*, denoted $\mathrm{VD}_k(L)$, subdivides $\mathbb{R}^3$ into maximal regions that have the same k nearest lines. The case $k = n-1$ gives the *farthest Voronoi diagram*, denoted $\mathrm{FVD}(L)$. Each diagram is a 3D cell complex, whose *features* are vertices, edges, faces (2D), and cells (3D). Unless otherwise stated, we call 2D faces simply *faces* and 3D cells simply *cells*.

The general position assumption implies the following properties. By assumption (1), the topology of trisectors is unique, [12] Theorem 1. By (2), related trisectors intersect transversely (see Lemma 3). By (3), vertices of the nearest and farthest Voronoi diagrams have degree 4. By (4), the asymptotes of related trisectors do not coincide (see Lemma 4).

▶ **Lemma 3.** *If two related trisectors (associated with four lines) intersect tangentially, then there exists a sphere tangent to the four lines such that the four contact points are coplanar.*

For a cell complex M, let B be a ball large enough to intersect any i-dimensional feature of M, where $0 \leq i \leq 3$, in one connected component. Let Γ be the boundary of B. The intersection $M \cap \Gamma$ encodes the unbounded features of M into a two-dimensional map on Γ, denoted $\Gamma(M)$. In particular, the unbounded edges, unbounded 2D faces, and unbounded 3D cells of M correspond exactly to the vertices, edges and faces, respectively, of $\Gamma(M)$.

Next, we give a summary of known elementary properties for the Voronoi diagram of four lines. Four lines induce four trisectors and six bisectors. When two trisectors intersect, the other two trisectors necessarily pass through the same intersection points. These intersections are Voronoi vertices in the order-k Voronoi diagram, $k \in \{1, 2, 3\}$. Voronoi vertices lie on all six bisectors. A trisector consists of four unbounded curves, called branches. Trisector branches are partitioned by the Voronoi vertices into arcs, which are order-k Voronoi edges for some k. Each trisector lies in three bisectors, and each bisector contains exactly two trisectors. A bisector is partitioned by the two trisectors it contains into faces, where each face belongs to an order-k Voronoi diagram, for some k. The nearest and farthest Voronoi diagrams of four lines share no edges or faces, by definition. Instead, both diagrams share their edges with the order-2 Voronoi diagram.

2.1 Algebraic framework and geometric reinterpretation

We recall the parametrization of lines from [12]. WLOG, let the coordinate system satisfy the following: ℓ_1 is defined by point $(0, 0, 1)$ and vector $(1, a, 0)$, ℓ_2 is defined by point $(0, 0, -1)$ and vector $(1, -a, 0)$, for some $a \in \mathbb{R}$. For $i \geq 3$, line ℓ_i is defined by point $(b_i, c_i, 0)$ and vector $(d_i, e_i, 1)$, with $b_i, c_i, d_i, e_i \in \mathbb{R}$. Then, the bisector $B(\ell_1, \ell_2)$ has the following formulation:

$$B(\ell_1, \ell_2) = \{(x, y, z) \mid (x, y) \in \mathbb{R}^2, z = \frac{-a}{1 + a^2} \cdot xy\}.$$

There is a natural homeomorphism $\Pi : B(\ell_1, \ell_2) \to \mathbb{R}^2$, given by $\Pi(x, y, z) = (x, y)$, that projects the bisector onto the Euclidean plane. We call $\Pi(B(\ell_1, \ell_2))$ the *projected bisector*. For any trisector $T = T(\ell_1, \ell_2, \ell_i)$ on $B(\ell_1, \ell_2)$, we call $\Pi(T)$ the *projected trisector*. The

following lemma summarizes key properties of the Voronoi diagram of three lines from [12], including Propositions 14 and 17 of [12]. We restate them in the following lemma in terms of projected trisectors on projected bisectors, as needed for analyzing four lines in Sections 4 and 5. Refer to Figure 1.

▶ **Lemma 4** ([12]). *A projected trisector T consists of four unbounded branches. It has two vertical and two horizontal asymptotes. There is a unique branch that admits exactly one asymptote, called the* middle branch*. The middle branch partitions a projected bisector into two regions: one containing a single branch and the other containing two branches; denote the single branch as the* U branch*. Furthermore, the following hold:*

1. *the middle branch of T is the same on all three bisectors that T lies on;*
2. *the three branches of T, other than the middle branch, each becomes a U branch on exactly one of the three bisectors that T lies on;*
3. *the four branches of T partition the bisector into five faces, two of which belong to the* NVD *and three belong to the* FVD *of the three lines;*
4. *if the asymptote of the middle branch is vertical, then all branches are y-monotone; otherwise, all branches are x-monotone.*

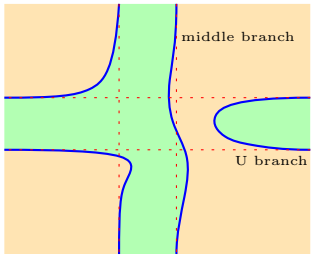


Figure 1 A projected trisector on a projected bisector. Branches are shown in blue and the asymptotes are shown in red. The NVD faces are shown in green and the FVD faces in orange.

3 Some combinatorial results on Voronoi diagrams of lines in $\mathbb{R}^3$

We first analyze the asymptotes of a projected trisector, building upon the results of [12]. The following lemma uses the parametrization notation of Section 2.1.

▶ **Lemma 5.** *The vertical and horizontal asymptotes of a projected trisector $\Pi(T(\ell_1, \ell_2, \ell_i))$, on the projected bisector $\Pi(B(\ell_1, \ell_2))$, are roots of quadratic polynomials whose coefficients depend only on the parameters a, d_i, e_i that define the direction of the line ℓ_i. Consequently, the asymptotes of the projected trisector are invariant under translation of the line ℓ_i.*

The number of vertices in the Voronoi diagram of four lines is known to be at most 8 [18]. We extend this observation and show that the number of vertices is always even.

▶ **Lemma 6.** *The number of vertices in a Voronoi diagram of four lines in general position is always even, between 0 and 8, and all numbers can be attained.*

The farthest Voronoi diagram of a set L of n lines in general position is known to have n^2-n distinct 3D cells, which are all unbounded, and do not form tunnels [5]. The following lemma follows from [5] and extends this result to the cells of individual lines.

▶ **Lemma 7.** *In the farthest Voronoi diagram of n lines in general position, $n \geq 2$, the Voronoi region of each line consists of exactly $n-1$ unbounded 3D cells.*

In NVD(L), where L is a set of n lines in general position, 3D Voronoi regions are connected, and each region is topologically an infinite cylinder with two unbounded ends, assuming $n \geq 3$. Thus, we can derive the following results for NVD(L). The farthest counterpart follows from [5] Theorem 5.12.

▶ **Lemma 8.** *The map* $\Gamma(\text{NVD}(L))$ *has* $4n-4$ *vertices,* $6n-6$ *edges, and* $2n$ *faces. The map* $\Gamma(\text{FVD}(L))$ *has* $2n^2-2n-4$ *vertices,* $3n^2-3n-6$ *edges, and* n^2-n *faces. The numbers indicate the unbounded Voronoi edges, faces, and cells, respectively, in each diagram.*

Note that the *Gaussian map* on the sphere of directions from [5] can be regarded as the limit of the unbounded features map on Γ, as Γ goes to infinity. Combining Lemma 8 and some counting arguments, we derive the following result, which we use for $n = 4$ in Section 5.

▶ **Theorem 9.** *Let* V_N *(resp.* V_F*) denote the number of vertices of* NVD(L) *(resp.* FVD(L)*).*
- NVD(L) *has* $2V_N+2n-2$ *edges,* V_N+3n-3 *faces, and* n *cells.*
- FVD(L) *has* $2V_F+n^2-n-2$ *edges,* V_F+2n^2-2n-3 *faces, and* n^2-n *cells.*

4 Twists

From this section onward, we focus on a set L of four lines in general position. The four lines give rise to four trisectors that form the trisector system of L, and six bisectors.

▶ **Definition 10.** *A pair of trisector intersections* (v_1, v_2) *is called a* twist*, if there exist two trisectors* T_1, T_2*, that both have a single branch passing through* v_1, v_2 *consecutively (i.e., no other vertices exist between* v_1 *and* v_2 *on either* T_1 *or* T_2*); see Figure 2.*

▶ **Definition 11.** *A twist* (v_1, v_2) *is called a* full twist*, if there are exactly four trisector branches passing through* v_1 *and* v_2 *consecutively (with no other vertices between* v_1, v_2 *on any branch); see Figure 2 right. A twist is called a* partial twist*, if there are exactly two trisectors with a single branch passing through both* v_1 *and* v_2*, and two trisectors with exactly two branches each, one passing through* v_1 *and the other through* v_2*; see Figure 2 left.*



Figure 2 Left: a partial twist. Right: a full twist. Different colors denote different trisectors.

The proof of the following lemma is deferred to the next section.

▶ **Lemma 12.** *A twist is either a full twist or a partial twist.*

4.1 Full twists

We show that the Voronoi diagram (both nearest and farthest) exhibits a unique local structure around any full twist; see Figure 3 for an illustration.

▶ **Lemma 13.** *Among the four bounded trisector arcs incident to a full twist, exactly two lie in* NVD(L) *and the other two lie in* FVD(L)*, defining one bounded face in each diagram called a full-twist face. The arcs* alternate *between the nearest and the farthest diagrams, alternate in the sense that the two full-twist faces intersect each other.*

Proof sketch. Assume to the contrary that at least 3 bounded arcs incident to the full twist vertices belong to NVD(L) (resp. FVD(L)); then each pair of these arcs bounds a distinct face of NVD(L) (resp. FVD(L)). But three such faces incident to both vertices of the full twist create a bounded 3D cell in the NVD(L) (resp. FVD(L)), which is a contradiction. Hence, there are two bounded arcs in NVD(L) and two in FVD(L) incident to the full twist. Assume now that the arcs are not alternating. WLOG, let the red and green arcs in Figure 3 left be in the NVD, and consider the face F_1 (resp. F_2) bounded by the red and blue (resp. green and brown) arcs in Figure 3 middle. Both faces must belong to $\mathrm{VD}_2(L)$, as each is bounded by one edge in NVD(L) and one edge in FVD(L). However, they intersect in a curve that is not an edge of the diagram, see Figure 3 middle, deriving a contradiction. ◀

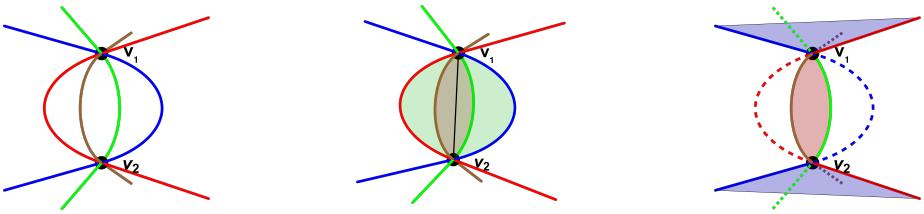


Figure 3 Left: a 3D view of a full twist. Middle: illustration for the proof of Lemma 13. Faces F_1 and F_2 are highlighted, intersecting at the black segment which is not an edge of the diagram. Right: the unique structure of the NVD(L) around the full twist.

The converse of the above lemma is also true, which will be shown in Lemma 16. Next, we describe a local modification on a Voronoi diagram associated with a full twist.

Local modification. Refer to Figure 4. Start with an NVD face incident to two edges from different trisectors (e.g., the blue face on $B(\ell_1, \ell_2)$ incident to the red and blue edges). Move the interior of the edges closer until they intersect twice (at vertices v_1, v_2). Consequently, each of the two original edges is split into three pieces, and the middle arcs no longer belong to the NVD. The original NVD face is split into two faces. Two new bounded edges from the other two trisectors appear in the diagram (the green and brown edges in Figure 4), which form a full-twist face in the NVD. The cells of ℓ_3 and ℓ_4 touch each other along the full-twist face. The local modification applies to the FVD in the same way.

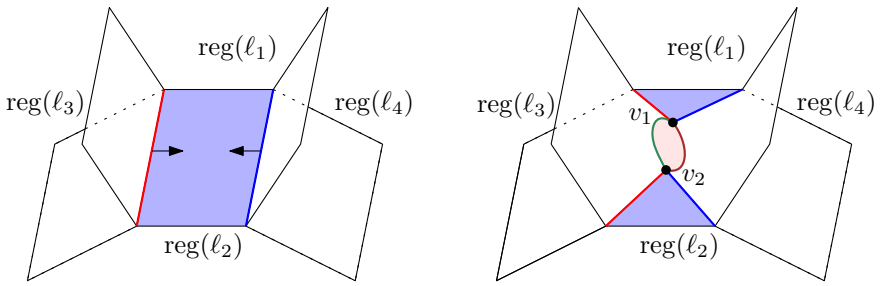


Figure 4 Left: local NVD/FVD structure; right: the modified diagram after adding a full twist.

We say that the diagram in Figure 4 right is obtained from the diagram in Figure 4 left by "adding a full twist". The local operation can be reversed by "removing a full twist". Since the effect of a full twist is purely local, we first focus on the global structure by restricting attention to Voronoi diagrams whose trisector arrangements have no full twists.

4.2 Partial twists and further properties

We illustrate the local structure of the Voronoi diagrams around twists, and prove that any twist in the trisector arrangement must be either full or partial.

▶ **Lemma 14.** *Let (v_1, v_2) be a twist and let (e_1, e_2) be the two trisector arcs incident to both v_1, v_2, as shown in Figure 5. Assume that a third trisector has two distinct branches, one of which passes through v_1 and the other through v_2. Then, one of e_1, e_2 belongs to* NVD(L) *and the other belongs to* FVD(L)*; the face bounded by e_1, e_2 belongs to $VD_2(L)$.*

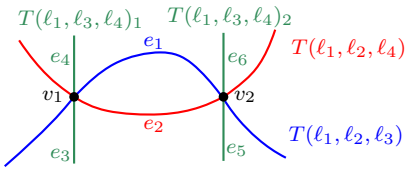


Figure 5 A local section of a trisector system, illustrating the setup of Lemma 14.

Proof. Refer to Figure 5. We distinguish two branches of trisector $T(\ell_1, \ell_3, \ell_4)$ as $T(\ell_1, \ell_3, \ell_4)_1$ and $T(\ell_1, \ell_3, \ell_4)_2$. Note that one of the edges e_3, e_4 in $T(\ell_1, \ell_3, \ell_4)_1$ is an NVD edge and the other is an FVD edge. Analogously for the edges e_5, e_6 in $T(\ell_1, \ell_3, \ell_4)_2$. Hence, WLOG, we assume that e_4, e_6 are edges of NVD(L).

Assume for the sake of contradiction that e_1, e_2 are both in NVD(L). Then, the face on $B(\ell_1, \ell_2)$ bounded by e_1 and e_2 is also in NVD(L). Further, there is a face of NVD(L) on $B(\ell_1, \ell_3)$ incident to e_1, e_4, e_6, and a face of NVD(L) on $B(\ell_1, \ell_4)$ incident to e_2, e_4, e_6. Hence, there are two faces f_1 and f_2 in NVD$(\{\ell_1, \ell_3, \ell_4\})$, on $B(\ell_1, \ell_3)$ and $B(\ell_1, \ell_4)$, respectively, that are incident to both branches $T(\ell_1, \ell_3, \ell_4)_1$ and $T(\ell_1, \ell_3, \ell_4)_2$. By [12], the Voronoi region of a line in the NVD of three lines is invariant, which is shown schematically in Figure 6-middle. To satisfy the aforementioned properties, faces f_1 and f_2 must be the top and bottom faces in Figure 6-middle, and the two branches $T(\ell_1, \ell_3, \ell_4)_1$ and $T(\ell_1, \ell_3, \ell_4)_2$ must be the green branches in the same figure. Adding line ℓ_2 to NVD$(\{\ell_1, \ell_3, \ell_4\})$ creates vertices v_1, v_2 on the green branches, with incident edges e_1, e_2 on the top and bottom faces, and a face bounded by e_1, e_2 in NVD(L), see Figure 6-right. This new face splits the 3D cell of ℓ_1 into two components, which is a contradiction as the 3D cells of NVD(L) are topological infinite cylinders.

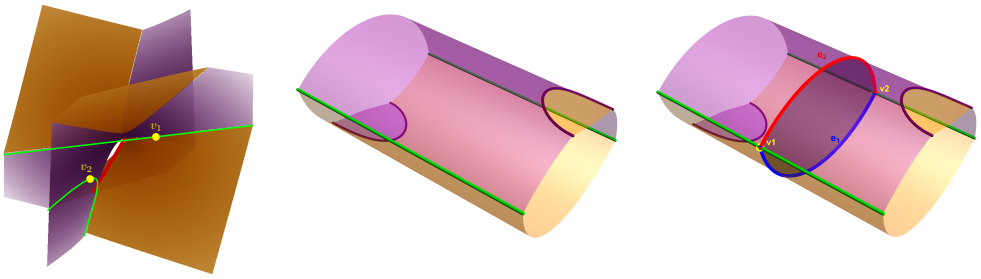


Figure 6 The region of ℓ_1 in NVD(ℓ_1, ℓ_3, ℓ_4), with the two green trisector branches highlighted. Left: a real instance. Middle: a schematic representation. The purple faces belong to $B(\ell_1, \ell_4)$ and the yellow faces belong to $B(\ell_1, \ell_3)$. Right: the result of inserting ℓ_2.

Next, assume that e_1, e_2 are both in FVD(L). Analogously, we can show that there are two faces in FVD($\{\ell_1, \ell_3, \ell_4\}$), on $B(\ell_1, \ell_3)$ and $B(\ell_1, \ell_4)$, respectively, that are incident to both trisector branches $T(\ell_1, \ell_3, \ell_4)_1$ and $T(\ell_1, \ell_3, \ell_4)_2$. However, such a pair of trisector branches does not exist, by Lemma 4 (items 2 and 3). Thus, one of e_1, e_2 is an NVD and the other is an FVD edge. Consequently, the face bounded by e_1 and e_2 lies in $\mathrm{VD}_2(L)$. ◀

▶ **Corollary 15.** *For any partial twist, the face bounded by the two trisector arcs incident to both vertices of the partial twist belongs to the order-2 Voronoi diagram $VD_2(L)$.*

We can now use Lemma 14 to prove Lemma 12.

Proof sketch of Lemma 12. Let (v_1, v_2) be a twist and assume that it is not partial nor full. If three trisector branches pass through v_1, v_2 consecutively, then the fourth trisector T_4, either (a) has two branches through v_1 and v_2, or (b) has a single branch through v_1 and v_2 but v_1, v_2 are not consecutive along this branch as otherwise (v_1, v_2) would create a full twist. In case (a) (Figure 7 left), by the pigeonhole principle, at least two of the three bounded edges incident to the twist both lie in NVD(L) or both lie in FVD(L). This contradicts Lemma 14. In case (b), consider a bisector containing T_4 together with any other trisector. Since trisector branches are unbounded and do not self-intersect, the number of vertices on T_4 between v_1 and v_2 must be even. The case with two such vertices is shown in Figure 7 right. We derive a contradiction by exploring faces on different bisectors. Additional cases might arise, e.g., two trisectors pass through v_1, v_2 consecutively and the other two have single branches passing through v_1, v_2 not consecutively. These are all ruled out by arguments analogous to case (b). ◀

Case (b) of the above proof shows that "nested full twists", as shown in Figure 7 right, are impossible. Consequently, full twists are either isolated or sequential.

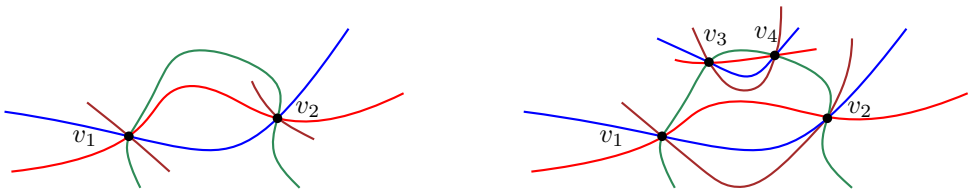


Figure 7 Left: a trisector system corresponding to (a) in the proof of Lemma 12. Right: two "nested full twists", corresponding to (b) in the proof of Lemma 12. Both are impossible.

▶ **Lemma 16.** *In* NVD(L) *(resp.* FVD(L)*), if there is a bounded face with exactly two edges and two vertices, then these vertices belong to a full twist.*

Proof sketch. If the two bounded edges both lie in NVD(L) (resp. FVD(L)), then each of the remaining two trisectors (other than those of the two edges) must have exactly one branch passing through the two vertices, by Lemma 14. If both remaining branches pass through the two vertices consecutively, then these vertices form a full twist. Otherwise, for at least one of the branches, there is an even number of vertices between the two vertices of the twist, which contradicts Lemma 12. ◀

Lemma 16 provides a concrete criterion to detect full twists by inspecting two projected trisectors (on a single bisector). It is used to classify Voronoi diagrams without full twists.

5 Identifying the Voronoi topologies by exhaustive search

In this section, we present an exhaustive search algorithm to classify the Voronoi diagram of four lines, assuming no full twists. To this end, we use the notion of a *configuration* as an abstraction for a projected bisector and the two relevant trisectors it contains.

We call a planar curve with four non-intersecting components *trisector-like*, if it satisfies the properties of the projected trisector of Lemma 4, in terms of the asymptotes and the middle trisector branch. Each trisector-like curve partitions the plane into faces, which we label as NVD or FVD according to Lemma 4, item 3. If a trisector-like curve is indeed the projected trisector of a set of three lines L, its faces are projections of $\mathrm{NVD}(L)$ and $\mathrm{FVD}(L)$.

We define a *configuration* on the plane as the overlay of two trisector-like curves. A configuration partitions the plane into faces, which we label as NVD, FVD, or VD_2. The faces labeled as NVD (resp. FVD) are the common intersections of the respectively labeled faces of the two trisector-like curves; any other face is labeled VD_2. A configuration is called *realizable* if there exist four lines L that define a bisector and two related trisectors whose projection has the same structure as the configuration. If a configuration is realizable, then the projected faces of $\mathrm{NVD}(L)$ (resp. $\mathrm{FVD}(L)$ and $\mathrm{VD}_2(L)$) on that bisector correspond exactly to the configuration faces labeled as NVD (resp. FVD and VD_2). For example, the configuration in Figure 8 right is realizable by the projected bisector and trisectors on the left.



Figure 8 Left: a real projected bisector with two projected trisectors on it. Right: a configuration. Green faces are NVD faces, orange faces are FVD faces, and white faces are VD_2 faces.

5.1 Properties of configurations

We give necessary conditions that restrict how trisectors and their asymptotes interact. These conditions rule out many unrealizable configurations. By Lemma 8, the map $\Gamma(\mathrm{NVD}(L))$ has 12 vertices that indicate the unbounded Voronoi edges.

▶ **Lemma 17.** *Given four lines L in general position, one of the following holds:*
1. *one trisector induces 6 vertices in $\Gamma(\mathrm{NVD}(L))$ and each other trisector induces 2;*
2. *one trisector induces 0 vertices in $\Gamma(\mathrm{NVD}(L))$ and each other trisector induces 4.*

Proof sketch. On a projected bisector, if there is an unbounded face in the positive x-direction, then there is also an unbounded face in the negative x-direction. These faces have two unbounded edges in each direction, which induce 4 vertices in $\Gamma(\mathrm{NVD}(L))$. If there are no such unbounded faces, then there are 0 vertices. The same holds for the y-directions. Hence, on one bisector, the two related trisectors together contribute 0, 4, or 8 vertices to $\Gamma(\mathrm{NVD}(L))$. The lemma follows by considering all six bisectors combined. ◀

A bisector is called an (i, j)-bisector if the two trisectors it contains induce i and j vertices in $\Gamma(\mathrm{NVD}(L))$. As a corollary of Lemma 17, we obtain the following classification of bisectors.

▶ **Corollary 18.** *Given four lines L in general position, one of the following holds:*
1. *there are three $(2,2)$-bisectors and three $(2,6)$-bisectors (with respect to $\Gamma(\mathrm{NVD}(L))$);*
2. *there are three $(0,4)$-bisectors and three $(4,4)$-bisectors (with respect to $\Gamma(\mathrm{NVD}(L))$).*

▶ **Lemma 19.** *Consider a bisector with two projected trisectors T_i, T_j. Let x_i^-, x_i^+ $(x_i^- < x_i^+)$ (resp. y_i^-, y_i^+) be the two vertical (resp. horizontal) asymptotes of T_i; analogously for T_j. It is impossible that simultaneously $x_i^- < x_j^- < x_j^+ < x_i^+$ and $y_j^- < y_i^- < y_i^+ < y_j^+$.*

Proof sketch. By Lemma 5, we express the asymptotes of the projected trisector $T(\ell_1, \ell_2, \ell_i)$ in the parameters a, d_i, e_i. Similarly for $T(\ell_1, \ell_2, \ell_j)$. We translate the given relation to a system of inequalities in the parameters and show that the system is infeasible. ◀



Figure 9 Left: a $(0,8)$-bisector, impossible by Corollary 18. Right: impossible by Lemma 19.

We apply Corollary 18 and Lemma 19 to filter out unrealizable configurations such as those illustrated in Figure 9. We say that two related trisectors (on a bisector) are *parallel* (denoted "$\cdot \parallel \cdot$"), if they are both x-monotone or both y-monotone; otherwise, they are not parallel ("$\cdot \nparallel \cdot$"). Parallelism of trisectors is transitive. The following is used as a filter in Section 5.2.

▶ **Lemma 20.** *For any three pairwise related trisectors T_i, T_j, T_k, if $T_i \parallel T_j$ and $T_j \parallel T_k$, then $T_i \parallel T_k$; if $T_i \nparallel T_j$ and $T_j \nparallel T_k$, then $T_i \parallel T_k$; see Figure 10.*

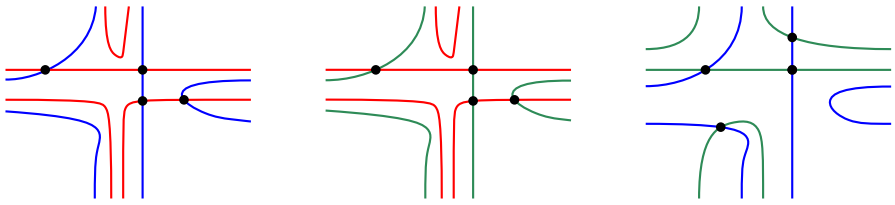


Figure 10 Three bisectors and trisectors; $T(\ell_1, \ell_2, \ell_3)$ is red, $T(\ell_1, \ell_2, \ell_4)$ is blue, $T(\ell_1, \ell_3, \ell_4)$ is green. They are not realizable by Lemma 20.

Next, we consider 6-*tuples of configurations*, where each configuration corresponds to one bisector associated with four lines. A configuration tuple is realizable if and only if there exists a set of four lines that realizes every configuration in the tuple. In a realizable 6-tuple, each trisector-like curve (12 in total) represents a trisector $T(\ell_i, \ell_j, \ell_k)$ whose representation must be consistent with all the elementary properties listed in Section 2 and Lemma 4 (by viewing a configuration as a bisector and a trisector-like curve as a trisector). Note that there are exactly three trisector-like curves that represent $T(\ell_1, \ell_2, \ell_3)$, and they lie on the configurations of $B(\ell_1, \ell_2)$, $B(\ell_1, \ell_3)$, and $B(\ell_2, \ell_3)$.

5.2 A search algorithm, proving Theorems 1 and 2

To classify the topology of the Voronoi diagram of four lines without full twists, we use the following exhaustive search algorithm.

Phase 1: Generate configurations. We generate all configurations with at most 8 vertices, excluding those that lead to full twists. Phase 1 works in two steps. In Step 1, we generate simple configurations that have no twists. By Lemma 4, such a configuration is uniquely determined by the asymptotes and the choice of the middle branches. We construct the simple configurations by enumerating all asymptote arrangements and all choices of middle branches of the two trisector-like curves. We label the faces of each configuration as described above. In Step 2, we add twists to simple configurations, as many as possible, provided that the number of resulting vertices is at most 8. To add a twist, we select two configuration edges that bound an existing face, fix their endpoints, and intersect their interior twice more. To avoid creating full twists, we only consider edge pairs incident to a face with label VD_2 (see Figure 11 (a)-(b)). This is sufficient because intersecting any other pair of edges produces a full twist, as shown in the proof of Theorem 1.

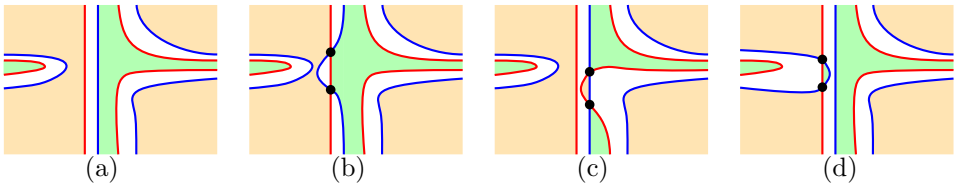


Figure 11 (a) A simple configuration. Faces labeled NVD (resp. FVD) are shown in green (resp. orange). (b) Add a twist by intersecting two edges bounding a VD_2 face (in white). (c), (d) Add a twist by intersecting two edges bounding an NVD or FVD face; both lead to a full twist.

Phase 2: Filter out configurations. We filter out any configurations of Phase 1 that are not realizable by applying filters derived from Corollary 18 and Lemma 19.

Phase 3: Filter out unrealizable configuration tuples. We generate all possible configuration 6-tuples from the configurations that remain at the end of Phase 2. We filter out unrealizable tuples by applying filters derived from the elementary properties of Section 2, Corollary 18, Lemma 20, , and Theorem 9, and obtain the set $\mathcal{P}$ of configuration tuples that survive these filters. Since the search space is large, we implement the filters in a program [22], which outputs 15 remaining configuration tuples at the end of Phase 3.

Each surviving configuration tuple in $\mathcal{P}$ can be realized by a set of four lines. It corresponds to a unique Voronoi diagram topology, as shown in the proof of Theorem 1. Then, straightforward inspection of the tuples and their topologies reveals the following properties.

▶ **Observation 21.** *Assuming no full twists, the following hold for* $\mathrm{NVD}(L)$ *and* $\mathrm{FVD}(L)$.

1. *Topologies with 0 or 2 vertices have no partial twists in their trisector system.*
2. *Topologies with 4 or more vertices have at least one partial twist.*
3. *No topology has 8 vertices; hence 8 Voronoi vertices require the presence of full twists.*
4. *The 2D faces of both* $\mathrm{NVD}(L)$ *and* $\mathrm{FVD}(L)$ *are all unbounded.*
5. *Each distinct* FVD *topology has a distinct map of unbounded features* $\Gamma(\mathrm{FVD}(L))$.
6. *There are only two distinct maps* $\Gamma(\mathrm{NVD}(L))$, *each corresponding to a case of Lemma 17.*

7. *A partial twist involves at least two middle trisector branches, each incident to exactly one vertex of the partial twist.*
8. *The 1-skeleton of* NVD(L) *has a single connected component that contains all vertices. This does not hold for* FVD(L).
9. *The edges of* FVD(L) *are unbounded except those incident to both vertices of a partial twist.*

Proof sketch of Theorem 1. We first argue that Phase 1 generates all possible configurations that do not induce full twists. Step 1 generates all simple configurations without twists. In Step 2, consider two arbitrary edges e_1, e_2 in a simple configuration. If e_1 and e_2 are incident to a common face labeled NVD (resp. FVD), then intersecting them twice creates a bounded face with two vertices labeled FVD (resp. NVD), see Figure 11 (c) and (d). By Lemma 16 such a face yields a full twist.

Assume now that e_1 and e_2 are not incident to a common configuration face. To intersect them, there must be at least one intermediate edge e_3 such that we first intersect e_1 and e_3 twice, after which e_1 and e_2 may become adjacent so that they can intersect. But the double intersections between e_1 and e_3 are no longer consecutive along e_1, so there is only one twist possible, namely the twist between e_1 and e_2 after they become adjacent. By a generalization of Lemma 16, the twist between e_1 and e_2 is a full twist. The case of more than one intermediate edge between e_1 and e_2 can be handled analogously. Hence, Step 2 correctly considers edges that are only incident to faces labeled VD_2, and these are enough to generate all relevant configurations.

Phases 2 and 3 use filters derived from necessary conditions of the diagrams that have already been proved. Thus, the output set $\mathcal{P}$ of Phase 3 contains every configuration tuple that may be realizable. Examples verify that the configuration tuples in $\mathcal{P}$ are realizable.

Each configuration tuple fixes the vertices, edges, and faces of all six configurations. The features with label NVD glue together uniquely, since the 2-skeleton of the Voronoi diagram of lines is connected. Analogously for the features with label FVD. Thus, there are at most 15 possible topologies for each of NVD(L) and FVD(L), which are in one-to-one correspondence with the configuration 6-tuples, and thus they are in one-to-one correspondence to each other. The correspondence naturally extends to $\mathrm{VD}_2(L)$. The remaining assertions of the theorem follow from inspecting the 15 configuration tuples. This completes the proof. ◀

As already mentioned, all configuration tuples that survive Phase 3 are realizable. This is crucial for the proof of Theorem 2, and further implies that the necessary conditions of the filters are also sufficient.

Proof sketch of Theorem 2. Consider four lines and their four trisectors. If there are no full twists in the trisector system, then the Voronoi diagram falls into one of the 15 topologies of Theorem 1. Assume that there are some full twists; by Lemma 12 they cannot be nested. Any full twist can be removed locally by the reverse local modification described in Section 4.1. Furthermore, if a configuration tuple satisfies the filters of the exhaustive search algorithm, then the configuration tuple derived after the removal of a full twist also does. Hence, the resulting configuration tuple, after removing all full twists, is one of the 15 configuration tuples in $\mathcal{P}$, all of which are realizable. This completes the proof. ◀

6 Examples and concluding remarks

Example 1: Voronoi diagram of four lines with 0 vertices. Assuming that there are 0 vertices, there are only two configurations (type (a) and type (b)) that satisfy Corollary 18 and Lemma 19; see Figure 12. Only one 6-tuple of configurations satisfies Corollary 18: three configurations of type (a) and three of type (b).



type (a) type (b)

Figure 12 Two configurations with 0 vertices. Faces with label NVD, FVD, and VD_2 are in green, orange, and white, respectively. Type (a) gives a $(2,6)$-bisector and type (b) gives a $(2,2)$-bisector.

An NVD example is shown in Figure 13 left. Of the four Voronoi regions, one is an unbounded triangular prism with three side faces (the region of the orange line in Figure 13 left); the other three have identical structure, which is schematically illustrated in Figure 13 right. The farthest Voronoi diagram is depicted schematically in Figure 15 left.

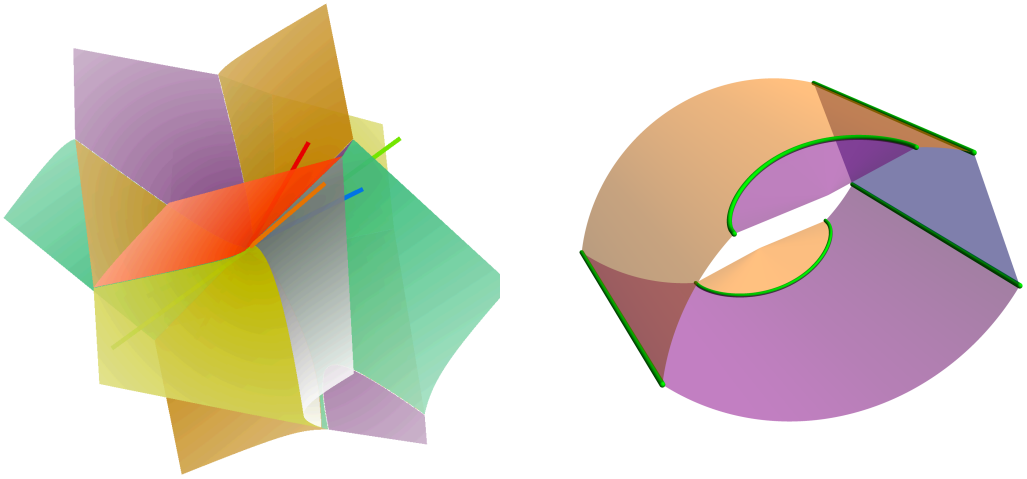


Figure 13 Left: NVD(L) with 0 vertices, faces in the same color are from the same bisector. Right: schematic illustration of a Voronoi region of NVD(L). Compare this with a region in the Voronoi diagram of three lines shown in Figure 6 middle.

Example 2: Voronoi diagram of four lines with 6 vertices. We show an example where the 6-tuple of configurations is symmetric: it consists of three configurations of type (c) and three of type (d), shown in Figure 14. The FVD is shown schematically in Figure 15 right.

Our classification results yield an algebraically simple method to compute the structure of the Voronoi diagram of four lines. We start by computing the Gaussian map of FVD(L) [5], which is simple algebraically. We then extract $\Gamma(\text{FVD}(L))$ by a minor adjustment (concerning the so-called vertices of anomaly of [5]). From $\Gamma(\text{FVD}(L))$, we can directly identify the matching base topologies of NVD(L) and FVD(L). Next, we decide if full twists exist by comparing the number of vertices in the base topology with the number of projected trisector intersections. The latter equals the number of real roots of a degree 8 univariate polynomial, which we compute via Sturm's theorem. If equal, there are no full twists, and the base topology provides the answer; otherwise, we need to locate the full twists. To this end, we derive the real configurations of each bisector (by applying the separation of four branches of

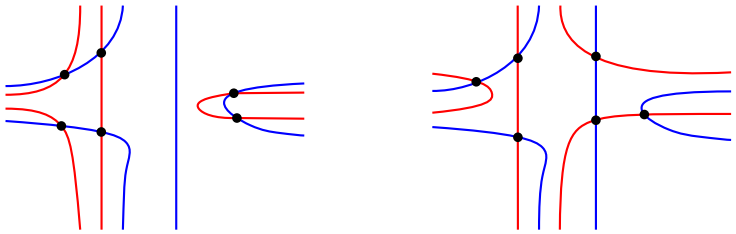


Figure 14 Two configuration types involved in Example 2.

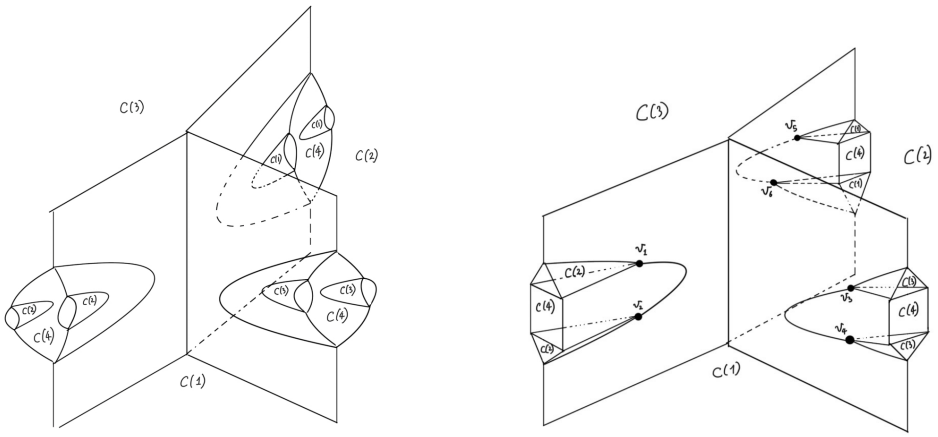


Figure 15 Left: schematic of the FVD(L) with 0 vertices. Right: schematic of the FVD(L) with 6 vertices of Example 2. Each label $C(i)$ denotes a cell in the farthest region of ℓ_i.

a trisector given in [12] to obtain (1) the corresponding simple configurations as in Phase 1, and (2) the number of intersections between all pairs of trisector branches). We compare with the configurations of the base topology, revealing the vertices of full twists, and add them to the base topology via the local modification. All operations can be done in low algebraic degrees, the most expensive being the location of full twists (whose degree is in the range of 8).

In conclusion, we have obtained a complete characterization of Voronoi diagrams of four lines in general position in $\mathbb{R}^3$. This offers a concrete foundation for future research on the Voronoi diagrams of n lines.

References

1 Pankaj K Agarwal, Boris Aronov, and Micha Sharir. Computing envelopes in four dimensions with applications. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 348–358, 1994. doi:10.1145/177424.178081.

2 Boris Aronov. A lower bound on Voronoi diagram complexity. *Information Processing Letters*, 83(4):183–185, 2002. doi:10.1016/S0020-0190(01)00336-2.

3 Franz Aurenhammer, Bert Jüttler, and Günter Paulini. Voronoi Diagrams for Parallel Halflines and Line Segments in Space. In Yoshio Okamoto and Takeshi Tokuyama, editors, *28th International Symposium on Algorithms and Computation (ISAAC 2017)*, volume 92 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:10, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2017.7.

4 Franz Aurenhammer, Evanthia Papadopoulou, and Martin Suderland. Piecewise-Linear Farthest-Site Voronoi Diagrams. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, volume 212 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:11, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2021.30.

5 Gill Barequet, Evanthia Papadopoulou, and Martin Suderland. Unbounded regions of high-order Voronoi diagrams of lines and line segments in higher dimensions. *Discrete & Computational Geometry*, pages 1–29, 2023.

6 Jean-Daniel Boissonnat, Micha Sharir, Boaz Tagansky, and Mariette Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 79–88, 1995. doi:10.1145/220279.220288.

7 Bernard Chazelle. An optimal convex hull algorithm and new results on cuttings (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico*, pages 29–38. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185345.

8 L Paul Chew, Klara Kedem, Micha Sharir, Boaz Tagansky, and Emo Welzl. Voronoi diagrams of lines in 3-space under polyhedral convex distance functions. *Journal of Algorithms*, 29(2):238–255, 1998. doi:10.1006/JAGM.1998.0957.

9 Tamal K Dey and Wulue Zhao. Approximate medial axis as a Voronoi subcomplex. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 356–366, 2002. doi:10.1145/566282.566333.

10 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1:25–44, 1986. doi:10.1007/BF02187681.

11 Michal Etzion and Ari Rappoport. Computing the Voronoi diagram of a 3-d polyhedron by separate computation of its symbolic and geometric parts. In *Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 167–178, 1999. doi:10.1145/304012.304029.

12 Hazel Everett, Daniel Lazard, Sylvain Lazard, and Mohab Safey El Din. The Voronoi diagram of three lines. *Discrete & Computational Geometry*, 42(1):94–130, 2009. doi:10.1007/S00454-009-9173-3.

13 Marc Glisse. Lower bound on the Voronoi diagram of lines in $\mathbb{R}^d$, 2021. arXiv:2103.17251.

14 Michael Hemmer, Ophir Setter, and Dan Halperin. Constructing the exact Voronoi diagram of arbitrary lines in three-dimensional space: with fast point-location. In *European Symposium on Algorithms*, pages 398–409. Springer, 2010. doi:10.1007/978-3-642-15775-2_34.

15 Christian Icking and Lihong Ha. A tight bound for the complexity of Voroni diagrams under polyhedral convex distance functions in 3d. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 316–321, 2001.

16 Victor Klee. On the complexity of d-dimensional Voronoi diagrams. *Archiv der Mathematik*, 34(1):75–80, 1980.

17 Vladlen Koltun and Micha Sharir. Polyhedral Voronoi diagrams of polyhedra in three dimensions. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 227–236, 2002. doi:10.1145/513400.513428.

18 Vladlen Koltun and Micha Sharir. Three dimensional Euclidean Voronoi diagrams of lines with a fixed number of orientations. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 217–226, 2002. doi:10.1145/513400.513427.

19 Joseph S. B. Mitchell and Joseph O'Rourke. Computational geometry column 42. *International Journal of Computational Geometry & Applications*, 11(5):573–582, 2001. doi:10.1142/S0218195901000651.

20 Raimund Seidel. On the number of faces in higher-dimensional Voronoi diagrams. In *Proceedings of the third annual symposium on Computational geometry*, pages 181–185, 1987. doi:10.1145/41958.41977.

21 Micha Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. *Discrete & Computational Geometry*, 12:327–345, 1994. doi:10.1007/BF02574384.

22 Zeyu Wang. Search engine for the Voronoi diagram of four lines in $\mathbb{R}^3$, March 2026. doi:10.5281/zenodo.19090548.

ETH-Tight Complexity of Optimal Morse Matching on Bounded-Treewidth Complexes

Geevarghese Philip
Chennai Mathematical Institute, India
CNRS IRL ReLaX, Chennai, India

Erlend Raa Vågset
Western Norway University of Applied Sciences (HVL), Førde, Norway

Abstract

The Optimal Morse Matching (OMM) problem asks for a discrete gradient vector field on a simplicial complex that minimizes the number of critical simplices. It is NP-hard and has been studied extensively in heuristic, approximation, and parameterized complexity settings. Parameterized by treewidth k, OMM has long been known to be solvable on triangulations of 3-manifolds in $2^{O(k^2)}n^{O(1)}$ time and in FPT time for triangulations of arbitrary manifolds, but the exact dependence on k has remained an open question. We resolve this by giving a new $2^{O(k \log k)}n$-time algorithm for any finite regular CW complex, and show that no $2^{o(k \log k)}n^{O(1)}$-time algorithm exists unless the Exponential Time Hypothesis (ETH) fails.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Algorithm design techniques; Mathematics of computing → Combinatorial algorithms; Mathematics of computing → Algebraic topology

Keywords and phrases Discrete Morse Theory, Simplicial Complexes, Optimal Morse Matching, Treewidth, Parameterized Algorithms, Computational Topology, Dynamic Programming, Exponential Time Hypothesis, Topological Data Analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.85

Related Version *Full Version*: `https://arxiv.org/abs/2603.05406` [39]

Funding *Erlend Raa Vågset*: Supported in part by the Research Council of Norway, grant "Parameterized Complexity for Practical Computing (PCPC)" (No. 274526).

Acknowledgements Geevarghese Philip thanks Priyavrat Deshpande for introducing him to Morse matchings and to questions concerning their efficient computation.

1 Introduction

Classical Morse theory [37] and its discrete counterpart [21, 22] provide a framework for simplifying spaces while preserving their essential topological features (see Figure 1). They do so by relating scalar functions to gradient flows in the smooth setting and discrete Morse functions to discrete gradient vector fields (Morse matchings) in the combinatorial setting. Much of the computational work on discrete Morse theory has focused on the matching perspective [32, 34, 40, 27, 1, 23], with the pioneering work on treewidth in computational topology [12] being no exception. At first sight, this close relationship suggests that the two formulations are interchangeable. In reality, they form diverging roads in the algorithmic landscape; let us follow the one less traveled by.

Morse theory has applications in topological data analysis and computational topology [27, 2, 33, 38, 23, 16], robotics and configuration spaces [20, 24], molecular modeling [14], and mesh and image processing [35, 18]. A recurring primitive in these works is to construct a discrete gradient vector field, also known as a Morse matching, with few critical simplices, yielding strong homotopy-preserving simplifications. Intuitively, such a field is a discrete

© Geevarghese Philip and Erlend Raa Vågset;
licensed under Creative Commons License CC-BY 4.0

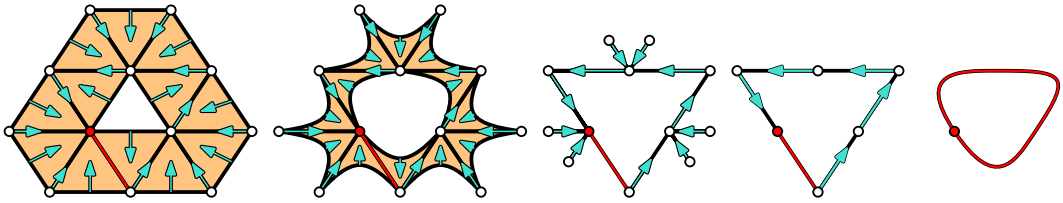


Figure 1 Discrete Morse theory can simplify a space while preserving its homotopy type.

vector field without "swirls": there are no loops in the flow lines, so one can contract the space along them without changing the homotopy type. Combinatorially, such a field corresponds to a matching in the Hasse diagram of the complex such that reversing the matched edges does not create any directed cycles. This motivates the Optimal Morse Matching (OMM) problem, also known as Min-/Max-Morse Matching, depending on whether the objective is to minimize the number of critical simplices or maximize the number of matched pairs:

▶ **Problem 1** (Optimal Morse Matching (OMM)).

Input: A finite simplicial complex K and a weight function $\omega : K \to \mathbb{R}_{\geq 0}$.
Task: Find a discrete gradient vector field W on K.
Optimize: Minimize the total weight of critical (i.e., unmatched) simplices.

From a computational complexity perspective, OMM and closely related variants are highly intractable: they are NP-hard [32], admit strong classical inapproximability bounds [3], and are linked to other hard topological decision problems such as Collapsibility [42, 36], Erasibility of 2-complexes [12, 3], and Shellability [25]. Algorithmic work includes approximation algorithms for Max-Morse Matching [40] and a variety of heuristic and workflow-based methods for practical simplification [2, 34, 27, 23, 1]. On the parameterized side, using the optimum number of critical simplices as parameter, Erasability and Min-Morse Matching are W[P]-hard and admit no FPT approximation schemes [12, 4].

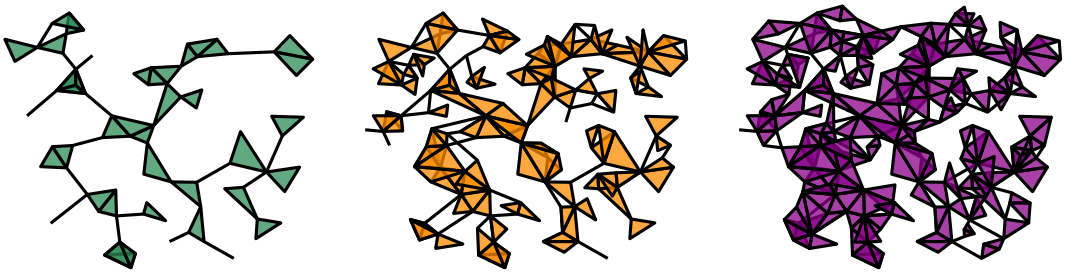


Figure 2 Three spaces from left to right, all of relatively low (but increasing) treewidth.

To circumvent these complexity barriers we turn to treewidth, which intuitively measures how close a combinatorial object is to being a tree; see Figure 2. An early systematic use of treewidth in discrete Morse theory is the fixed-parameter algorithm of [12] for OMM, which runs in time $2^{O(t^2)}n^{O(1)}$ when parameterized by the treewidth t of an associated spine graph for 2-complexes, and by the treewidth of either the spine graph or the dual graph for triangulated 3-manifolds. This was complemented by a Courcelle-style metatheorem for MSO-definable properties on triangulations [11], which yields fixed-parameter tractability for OMM on triangulated manifolds of fixed dimension when parameterized by the treewidth of

the dual graph. Since then, treewidth has become central in computational topology, leading to FPT algorithms for a range of NP-hard topological problems and invariants [10, 13, 6], ETH-tight lower bounds for several of these problems [8, 5, 7, 43], and detailed studies of the width of triangulated 3-manifolds [30, 28, 29].

▶ Remark 2. Across the literature, "treewidth k" is measured on different associated graphs (dual graphs, spine graphs, Hasse diagrams), chosen to suit the problem at hand. For fixed dimension, [11] shows that the treewidth of the Hasse diagram of a triangulation is bounded by a constant factor times the treewidth of its dual graph, and [12] shows that in dimension 3 the spine treewidth is likewise linearly bounded in terms of the dual treewidth. Thus, for fixed dimension these parameters coincide up to constant factors hidden in the $O(\cdot)$-notation. In this paper we simply write "treewidth k" and, in practice, work with the Hasse diagram.

1.1 Contributions

Algorithmic. We give an explicit dynamic program (DP) for a digraph formulation of Optimal Morse Matching parameterized by treewidth k, which in particular solves OMM on all finite regular CW complexes of treewidth k. It runs in time $2^{O(k \log k)}n$, thereby extending and improving both the earlier explicit treewidth-based algorithms for 2-complexes and triangulated 3-manifolds with running time $2^{O(k^2)}n^{O(1)}$ [12] and the implicit MSO-based algorithm for triangulated manifolds in [11]. The simple invariant and state space allowed us to implement the algorithm and verify it on small instances by exhaustive search.

Optimality. Under ETH, our running time is optimal: using a known treewidth-based lower bound for Directed Feedback Vertex Set (DFVS), we show that there is no $2^{o(k \log k)}n^{O(1)}$-time algorithm for Optimal Morse Matching parameterized by treewidth k, even on 2-dimensional complexes of top coface degree at most 4. This follows from a new polynomial-time reduction from DFVS to Erasability, to which we apply the recent Width Preserving Strategy (WiPS) framework of [43] to ensure that treewidth is preserved. Combined with the standard equivalence between Erasability and OMM on 2-dimensional complexes, this yields the ETH-tight $2^{\Theta(k \log k)}$ dependence.

Conceptual. Our order-based formulation (Feedback Morse Order) shows that working indirectly with vertex orders rather than directly with matchings captures exactly what a treewidth DP for OMM needs to remember. This ordering viewpoint transfers to Alternating Cycle-Free/Uniquely Restricted Matchings (AC-FM/URM) on bipartite graphs, but the connection breaks on general graphs. Here, AC-FM and URM still only admit $2^{\Theta(k^2)}n^{O(1)}$-time algorithms and lack tight-ETH lower bounds. We therefore point to AC-FM and URM as natural candidates for genuinely $2^{\Theta(k^2)}$-time problems in treewidth and as targets for either $2^{O(k \log k)}n$-time algorithms or matching $2^{o(k^2)}n^{O(1)}$ lower bounds.

▶ Remark 3. Some proofs of correctness, running-time analyses, and additional details are deferred to the full version [39].

2 Preliminaries

We use standard terminology from parameterized complexity [19, 17] and discrete Morse theory [21, 22, 41]. We write n for the input size; for OMM, n is the number of cells of the complex, which equals the number of vertices of its Hasse diagram.

2.1 Parameterized complexity theory

Treewidth and nice tree decompositions. When we speak of the treewidth of a digraph or a complex, we mean the treewidth of the natural underlying undirected graph in the first case, and of the Hasse diagram in the second. Informally, treewidth measures how close a graph is to being a tree (see also Figure 2): we cover the graph by overlapping bags of vertices arranged in a tree so that each edge lies entirely inside some bag and the bags containing any fixed vertex form a connected subtree; the width is one less than the size of the largest bag, and the treewidth k is the minimum width over all such decompositions; this k will be our parameter throughout the paper. Such decompositions let us localize global constraints: a dynamic program only needs to maintain partial solutions on a bag of size $k+1$ and combine them along the tree, so we can brute-force over states per bag rather than over the whole graph and obtain running times of the form $f(k)\,n^{O(1)}$. We use rooted nice tree decompositions; unless stated otherwise, we fix an arbitrary root bag and process the decomposition bottom–up. Formal definitions are given in the full version [39, App. A.1]. In this paper we work with the standard nice node types (leaf, introduce-vertex, forget-vertex, and join), and we also use auxiliary *introduce-edge* bags: unary nodes whose bag is identical to that of their child and that are annotated with a single edge uv in the bag, marking the point where that edge is processed in the dynamic program or reduction. This is a standard refinement that preserves width.

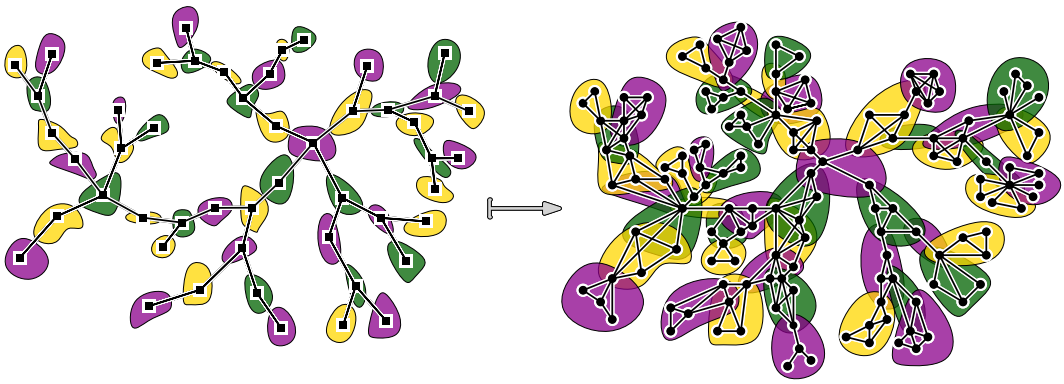


Figure 3 A graph G (right) and a tree decomposition T (left). Each node of T carries a bag $X_t \subseteq V(G)$, drawn as a blob containing vertices of G. Adjacent bags overlap so that every vertex and every edge of G is covered.

FPT and ETH. A parameterized problem with parameter k is *fixed-parameter tractable* (FPT) if it can be solved in time $f(k)\,n^{O(1)}$ for some computable function f; such running times are considered efficient when k is small compared to n, since the combinatorial explosion is confined to $f(k)$ while the dependence on the input size remains polynomial. For conditional lower bounds we assume the *Exponential Time Hypothesis* (ETH) [31], which states that 3-SAT on n variables cannot be solved in time $2^{o(n)}$; ETH is a strengthening of P $\neq$ NP that has withstood decades of algorithmic progress and underpins many tight running-time lower bounds. Our hardness source is Directed Feedback Vertex Set parameterized by the treewidth k of the underlying undirected graph of the input digraph D, where a *feedback vertex set* in a digraph $D = (V, E)$ is a set $J \subseteq V$ such that $D \setminus J$ is acyclic.

▶ **Problem 4** (Directed Feedback Vertex Set (DFVS)).

Input: A directed graph $D = (V, E)$ and an integer s.
Question: Does D contain a feedback vertex set of size at most s?

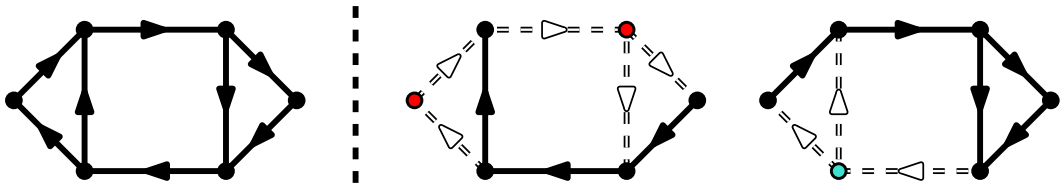


Figure 4 An instance of DFVS, a valid solution of size 2 and an optimal solution of size 1.

▶ **Theorem 5** (Bonamy et al. [9]). *Unless ETH fails,* DFVS *parameterized by the treewidth k of the underlying undirected graph of the input digraph cannot be solved in $2^{o(k \log k)} n^{O(1)}$-time.*

2.2 Discrete Morse theory

Complexes and Hasse diagrams. We use the standard notions of finite regular CW complexes and discrete Morse theory in the sense of Forman [21], and keep the topological preliminaries brief. Face relations are given by closure containment, so the face poset and its directed Hasse diagram encode the incidence data we use. We write $\overrightarrow{H}(X)$ for the directed Hasse diagram of a complex X; it has one vertex per cell and an arc $\sigma \to \tau$ whenever σ is an immediate face of τ (a cover relation in the face poset), and hence $\dim(\tau) = \dim(\sigma) + 1$. Let $H(X)$ be the underlying undirected graph; we measure treewidth on $H(X)$, denoted k. Simplicial complexes are the special case where cells are simplices (points, lines, triangles, tetrahedra, etc.); we use them in figures and in our reduction.

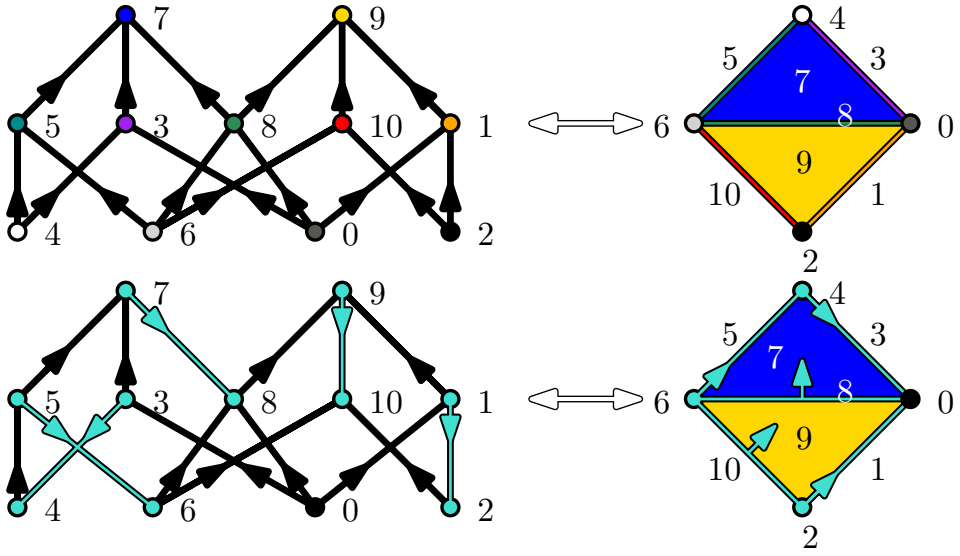


Figure 5 A discrete Morse function (top) on the Hasse diagram of a simplicial complex (left) and its geometric realization (right). Below, the induced discrete gradient vector field (Morse matching) is shown both on the Hasse diagram (left) and geometrically as a gradient vector field (right).

Discrete Morse theory on complexes. Forman's discrete Morse theory can be phrased entirely in terms of matchings on the Hasse diagram $\overrightarrow{H}(X)$ of a finite regular CW complex X. A *discrete vector field* is a matching W on $\overrightarrow{H}(X)$, i.e. a set of pairs (σ, τ) with $\sigma \to \tau$ an arc of $\overrightarrow{H}(X)$ such that each cell appears in at most one pair. From W we obtain a new

digraph $\overrightarrow{H}(X)_W$ by reversing exactly the matched arcs and leaving all others unchanged; if $\overrightarrow{H}(X)_W$ is acyclic, then W is a *discrete gradient vector field*. Forman's correspondence states that (i) every discrete Morse function on X induces such a gradient vector field, and (ii) conversely, every discrete gradient vector field arises from some discrete Morse function; in both directions, the unmatched cells in W are precisely the critical cells of the corresponding Morse function. Collapsing X along the gradient flow yields a smaller Morse complex (see Figure 1) that is homotopy equivalent to X and can make downstream topological computations much cheaper. The Optimal Morse Matching problem (Problem 1) asks for a discrete gradient vector field minimizing the total weight of these unmatched cells.

Erasibility in 2D. For our lower bound we use the notion of *erasibility* of 2-dimensional simplicial complexes K. A 1-simplex (edge) e is *free* if it is contained in exactly one 2-simplex τ, and removing e together with τ is an *elementary collapse*. A 2-simplex is *erasible* if it can be removed through a sequence of elementary collapses, and K is *erasible* if every 2-simplex can be eliminated in this way, that is, if K collapses to a 1-dimensional complex.

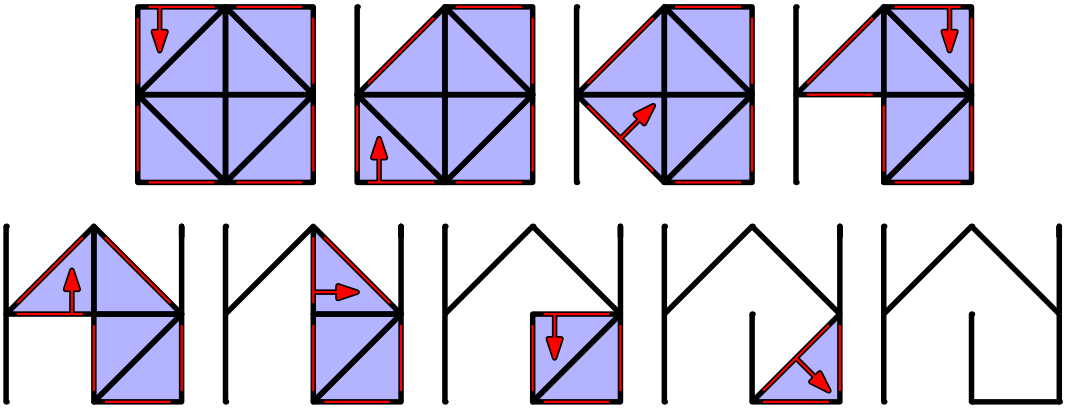


Figure 6 A triangulation of a square that is erasible with $S = \emptyset$: free edges are marked in red and elementary collapses by arrows. In contrast, a triangulation of a sphere has no free edges, so any erasibility requires $|S| > 0$.

▶ **Problem 6** (Erasibility).

Input: A 2-dimensional simplicial complex K and an integer $B \geq 0$.
Question: Is there a set S of 2-simplices in K such that $|S| \leq B$ and $K \setminus S$ is erasible?

▶ **Theorem 7** (Folklore; cf. [21, 32, 3]). *For finite 2-dimensional simplicial complexes,* Erasibility *and* Optimal Morse Matching *are computationally equivalent.*

3 Algorithm

We give a fixed-parameter algorithm for Feedback Morse Matching, a digraph generalization of Optimal Morse Matching, on digraphs whose underlying undirected graph has treewidth k. The algorithm rests on three ingredients: (i) we work in the general setting of arbitrary digraphs rather than Hasse diagrams only; (ii) we adopt the Morse-function viewpoint and encode solutions as vertex orders (Feedback Morse Orders) instead of matchings; and (iii) this yields a very simple dynamic program whose state at each bag

consists only of an order on the bag and a subset of its vertices. This order–mask formulation avoids the more involved connectivity machinery of previous approaches and leads to a running time of $2^{O(k \log k)}n$.

3.1 Of feedback Morse matchings and feedback Morse orders

Generalizing OMM to digraphs. For our algorithmic purposes it is convenient to generalize the matching viewpoint from Hasse diagrams to arbitrary digraphs, and to allow both positive and negative weights. Given a digraph $D = (V, E)$ and a matching $M \subseteq E$, let D_M be the digraph obtained by reversing every edge of M and leaving all other edges unchanged; we call M a *feedback Morse matching* if D_M is acyclic. In the resulting Feedback Morse Matching (FMM) problem (Problem 8), the input is a digraph D and a weight function $\omega : V(D) \to \mathbb{R}$, and the task is to find a feedback Morse matching minimizing the total weight of unmatched vertices. When D is the Hasse diagram $\overrightarrow{H}(K)$ of a simplicial complex K and $V(D)$ is identified with the simplices of K, discrete gradient vector fields on K are exactly feedback Morse matchings on D. Thus OMM is the special case of FMM where D comes from a complex and ω is nonnegative; in the classical unweighted case $\omega \equiv 1$, the objective is simply the number of critical simplices [32, 3]. Mixed-sign weights in this framework allow one to favour or penalize particular cells (useful for extending a Morse Matching), while the purely negative-weight variant (e.g. $\omega \equiv -1$) on general digraphs corresponds to finding a smallest feedback Morse matching, that is, a minimum-size matching whose reversals destroy all directed cycles in D.

▶ **Problem 8** (Feedback Morse Matching (FMM)).

Input: A finite directed graph $D = (V, E)$ and a weight function $\omega : V(D) \to \mathbb{R}$.
Question: Find a feedback Morse matching $M \subseteq E(D)$ that minimizes the total weight of unmatched vertices.

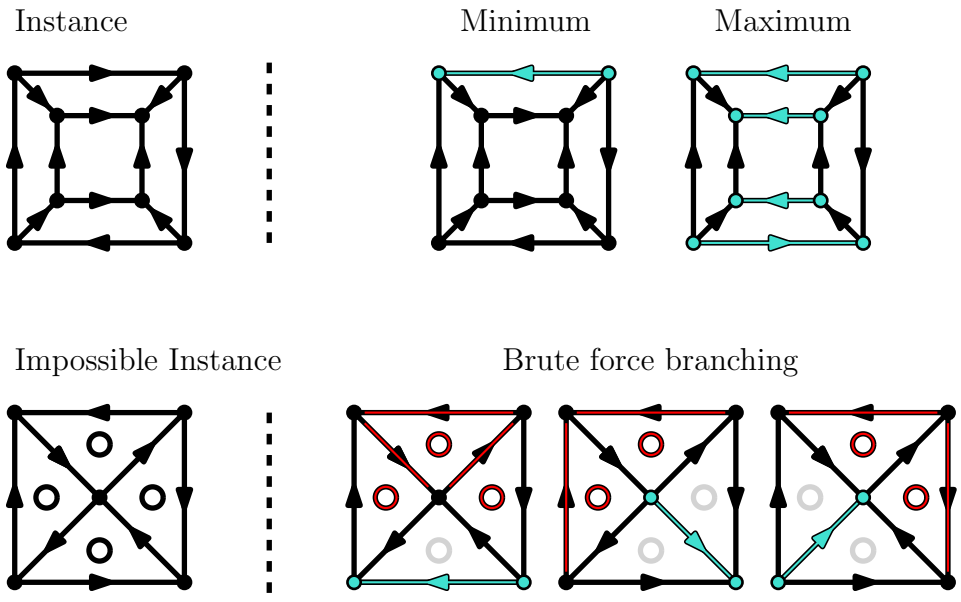


Figure 7 Feedback Morse matchings on digraphs. Top: an instance that admits feedback Morse matchings; two optimal solutions to FMM are shown. Bottom: an instance with no feedback Morse matching.

▶ **Theorem 9.** *Let $D = (V, E)$ be a digraph whose underlying undirected graph has treewidth k, and let $n := |V(D)|$. Given vertex weights $\omega : V \to \mathbb{R}$,* Feedback Morse Matching *on (D, ω) can be solved in time $2^{O(k \log k)} n$.*

Shifting from matchings to orders. As committed procrastinators, we now do our best to avoid thinking about matchings. Instead, we turn to vertex orders. Let $\pi = (v_1, \ldots, v_n)$ be a total order of V. An edge $(u, v) \in E$ is *backward* with respect to π if v appears before u in π, and let $M(\pi) := \{(u, v) \in E : (u, v)$ is backward w.r.t. $\pi\}$ be the set of backward edges. Once π is fixed, the set of edges to reverse is determined: we always take $M(\pi)$ and reverse exactly these edges. We call π a *feedback Morse order* if $M(\pi)$ is a matching (no two edges in $M(\pi)$ share a vertex).

▶ **Problem 10** (Feedback Morse Order (FMO))**.**

Input: A digraph $D = (V, E)$ and a weight function $\omega : V \to \mathbb{R}$.

Question: Find a feedback Morse order π that minimizes the total weight of vertices that are unmatched in $M(\pi)$.

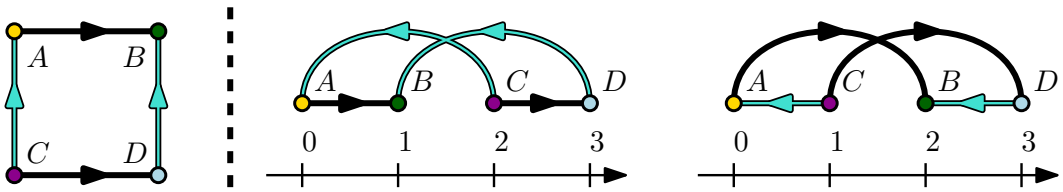


Figure 8 Two feedback Morse orders of the same digraph, drawn as permutations of the vertices. In both orders, the same set of backward edges (highlighted) forms a feedback Morse matching M, showing that one matching may admit many compatible orders; other orders can yield different backward-edge sets (and different matchings), or even no matching at all. For example, $D \prec C \prec B \prec A$ induces the other optimal matching $\{A \to B, C \to D\}$, $C \prec D \prec B \prec A$ induces only $\{A \to B\}$, and $A \prec D \prec C \prec B$ has backward edges $C \to A$ and $D \to C$, which is not a matching.

▶ **Lemma 11** (Matchings vs. orders)**.** *Let $D = (V, E)$ be a digraph. If π is a feedback Morse order, then $M(\pi)$ is a feedback Morse matching on D. Conversely, if M is a feedback Morse matching on D, then there exists a feedback Morse order π with $M(\pi) = M$.*

Proof. For any order π, reversing all backward edges makes every edge point forward along π, so reversing all backward edges yields an acyclic digraph; if $M(\pi)$ is a matching, it is a feedback Morse matching. Conversely, if M is a feedback Morse matching, then D_M is acyclic, and any topological order π of D_M satisfies $M(\pi) = M$. ◀

A Forman correspondence. By Lemma 11, FMO and FMM are equivalent optimization problems: every optimal feedback Morse order induces an optimal feedback Morse matching and vice versa, so in what follows we work entirely with the order-based formulation.

3.2 R-FMO: The boundary subproblem on bags

Orders and masks on bags. We run our dynamic program over a rooted nice tree decomposition of the underlying undirected graph of D, refined with introduce-edge bags as in Section 2. For a bag t let X_t be its vertex set, T_t the subtree rooted at t, W_t the vertices

appearing in bags of T_t, and G_t the subgraph on W_t whose edges are exactly those whose introduce-edge bags lie in T_t. Intuitively, G_t is the part of the graph already processed when we are at t, and X_t is its boundary. A global feedback Morse order π on D restricts at t to a total order g on X_t and a set $U \subseteq X_t$ of boundary vertices already matched inside G_t; we view U as a mask on X_t, marking which boundary vertices are already matched. This leads to the following boundary subproblem.

▶ **Problem 12** (Restricted Feedback Morse Order (R-FMO)).

Input: A digraph $G = (W, E')$ with vertex weights $\omega : W \to \mathbb{R}$, a boundary set $X \subseteq W$, a total order g of X, and a subset $U \subseteq X$.

Question: Among all feedback Morse orders π on G with $\pi|_X = g$ and $V(M(\pi)) \cap X = U$, minimize $\sum_{v \in W \setminus X : v \notin V(M(\pi))} \omega(v)$; if no such π exists, the optimum is defined as $+\infty$.

The global optimum of FMO (and hence of FMM) is the value of an R-FMO instance at the root bag, whose boundary is empty; in particular, every state in our dynamic program will consist of an order on the bag together with a mask of matched boundary vertices.

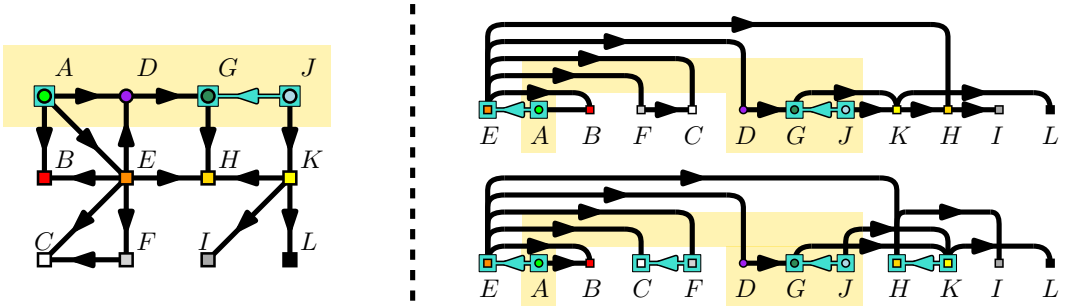


Figure 9 An instance of a Restricted Feedback Morse Order (R-FMO) subproblem (left), together with two feasible solutions (right): two different feedback Morse orders on the same instance (top and bottom) inducing different sets of matched and unmatched vertices.

3.3 Dynamic program on a tree decomposition

We now describe the dynamic program at a high level. Full recurrences and a formal DP invariant are presented in [39, App. A]. For each bag t and each state (g, U) on X_t as above, we maintain a table entry $c[t, g, U]$, defined as the optimum of the R-FMO instance (G_t, X_t, g, U), that is, the minimum total weight of unmatched vertices in $W_t \setminus X_t$ over all feedback Morse orders on G_t compatible with (g, U); if no such order exists, we set $c[t, g, U] = +\infty$. Figure 10 illustrates how such states are propagated along a tree decomposition and how locally invalid states are discarded. We process the nice tree decomposition bottom–up. At each bag type we update the table using only local information.

Leaf. The processed subgraph is empty, so there is a single state with empty order and empty matched set, and cost 0.

Introduce-vertex. A new vertex v enters the bag (and the graph) with no incident edges yet. It cannot already be matched, so $v \in U$ is forbidden. Otherwise we extend the order g by inserting v at its chosen position; the cost does not change.

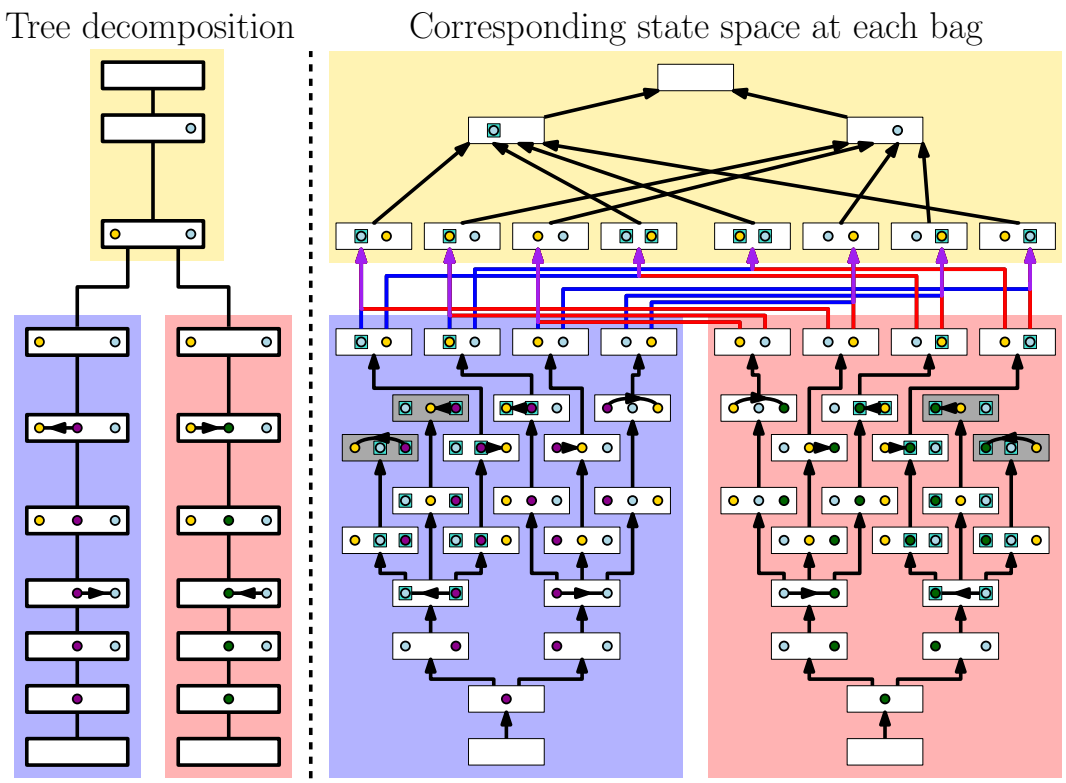


Figure 10 Dynamic program on the example from Figure 8. Bags of the tree decomposition are shown on the left; representative states (bag orders and matched subsets) on the right. States violating local adjacency, matching, or acyclicity constraints are discarded (grey).

Introduce-edge. A new edge (u, v) is introduced between vertices already in the bag. The order g determines whether it is forward or backward: (i) if (u, v) is forward in g, it can never be backward in any extension and thus can never enter the matching; (ii) if (u, v) is backward in g, it *must* be in the matching, and this is the unique place where it is introduced along the path. In the backward case we insist that u and v are currently unmatched in the child and become matched in the parent. Any state where this would create a double match in G_t is discarded.

Forget-vertex. A vertex v leaves the bag. At this moment all edges incident to v have already been introduced below, so v's matching status is final. We branch on whether v is matched in the child: if v is unmatched, we add $\omega(v)$ to the cost; if v is matched, v never contributes again. We then take the minimum over all child states in which v appears at some position in the child order and the projected order on the remaining bag is g.

Join. Two subtrees with the same bag X_t are merged. The processed subgraph G_t is the union of G_s and $G_{s'}$, and the bag order g is the same in all three bags. The order g already decides which bag-internal edges are backward, and hence which bag vertices are forced to be matched via those edges; call this set $M_I(g)$. Every feasible state $(g, U_\bullet)$ must satisfy $M_I(g) \subseteq U_\bullet$ for the parent and both children (if g induces conflicting backward edges, all such states are infeasible and $c[t, g, \cdot] = +\infty$). Outside $M_I(g)$, a bag vertex can be matched strictly below t in *at most one* of the two subtrees (there are no edges between the forgotten parts of

the two subgraphs). Thus for a parent state (g, U_t) we look over all pairs of child matched sets $(U_s, U_{s'})$ such that $M_I(g) \subseteq U_s, U_{s'}, U_t$ and $U_t \setminus M_I(g) = (U_s \setminus M_I(g)) \dot\cup (U_{s'} \setminus M_I(g))$. We then set $c[t, g, U_t]$ to the minimum of $c[s, g, U_s] + c[s', g, U_{s'}]$ over all such pairs.

A formal DP invariant and soundness/completeness proofs can be found in [39, App. A]. At the root bag r, the bag is empty, so there is a single state $(\emptyset, \emptyset)$; its value $c[r, \emptyset, \emptyset]$ is exactly the optimum of FMO on D, and hence of FMM and OMM.

3.4 Running time

We sketch the running-time bound; the full accounting is given in [39, App. A.11]. Let k be the treewidth of the underlying undirected graph of D and set $n := |V(D)|$. In a nice tree decomposition of width k, each bag has size at most $k{+}1$. A DP state is a pair (g, U) where g is a total order on the current bag and U is a subset of its vertices. Hence the number of states per bag is at most $(k{+}1)! \cdot 2^{k+1} \le (k{+}1)^{k+1} \cdot 2^{k+1} = 2^{(k+1)\log_2(k+1)+(k+1)} = 2^{O(k \log k)}$. For each fixed state, leaf/introduce-vertex/introduce-edge transitions take $k^{O(1)}$ time, and a forget transition branches over $k{+}2$ insertion positions. At a join bag, the order g is shared by both children, and combining solutions amounts to splitting the bag-mask information between the two subtrees, yielding at most 2^{k+1} admissible child pairs per state. Thus the work per bag is $2^{O(k \log k)}$, and since the decomposition has $O(n)$ bags, the total running time is $2^{O(k \log k)}\, n$.

4 ETH-optimality

How expensive is treewidth for OMM? Our $2^{O(k \log k)}n$-time algorithm shows what is achievable, and in this section we prove that this is indeed the true price under the Exponential Time Hypothesis (ETH). Starting from the ETH-based lower bound for Directed Feedback Vertex Set (DFVS) parameterized by treewidth, we give a new polynomial-time reduction from DFVS to Erasibility on 2-dimensional complexes of bounded coface degree. We then realize this reduction bag-by-bag using the Width Preserving Strategy (WiPS), which performs structural induction along a tree decomposition and keeps treewidth within a constant factor. As a consequence, any $2^{o(k \log k)}n^{O(1)}$-time algorithm for OMM (even in this restricted setting) would yield such an algorithm for DFVS, contradicting ETH and pinning down the dependence on k as $2^{\Theta(k \log k)}$.

▶ **Theorem 13.** *Assuming ETH,* Erasibility *parameterized by treewidth k admits no $2^{o(k \log k)}n^{O(1)}$-time algorithm, even when the input is restricted to 2-dimensional simplicial complexes of top coface degree at most* 4.

4.1 Gadgets and obstructions

Vertex gadgets: fuses and locks. Let D be a DFVS instance. For each vertex $v \in V(D)$ we build a local gadget $Y^v = \mathcal{F}(v) \cup \mathcal{L}(v)$, illustrated in Figure 11. The fuse $\mathcal{F}(v)$ is a 2-dimensional simplicial complex homeomorphic to a hollow cylinder $S^1 \times I$ with two boundary circles: one is a free boundary circle (marked in red) from which we can start collapsing the complex, "unravelling" the fuse by elementary collapses along the cylinder; the other boundary circle is non-free and is glued to the lock $\mathcal{L}(v)$. The lock $\mathcal{L}(v)$ is a 2-dimensional simplicial complex homeomorphic to a compact orientable surface of genus $\deg^+(v)$ with a single boundary component, which is attached to this non-free boundary circle of the fuse. For each outgoing arc $v \to w$ in D, we select a distinct simple closed curve on $\mathcal{L}(v)$ and glue it to a cross-section circle of the fuse $\mathcal{F}(w)$, so that each such curve acts like

a finger pinching the fuse: it prevents $\mathcal{F}(w)$ from collapsing past that cross-section as long as the lock $\mathcal{L}(v)$ is present. In the interior of $\mathcal{L}(v)$ we mark a single triangle deton(v) (the detonator); deleting this triangle makes the entire lock collapsible, eliminates all its pinches, and simultaneously releases every constraint that $\mathcal{L}(v)$ imposes on neighbouring fuses. We triangulate $\mathcal{L}(v)$ so that it becomes collapsible in either of two situations: after deleting deton(v), or after its attached fuse $\mathcal{F}(v)$ has been completely collapsed. In particular, as long as deton(v) is present and $\mathcal{F}(v)$ is intact, the lock cannot be collapsed while it still pinches some neighbouring fuse.

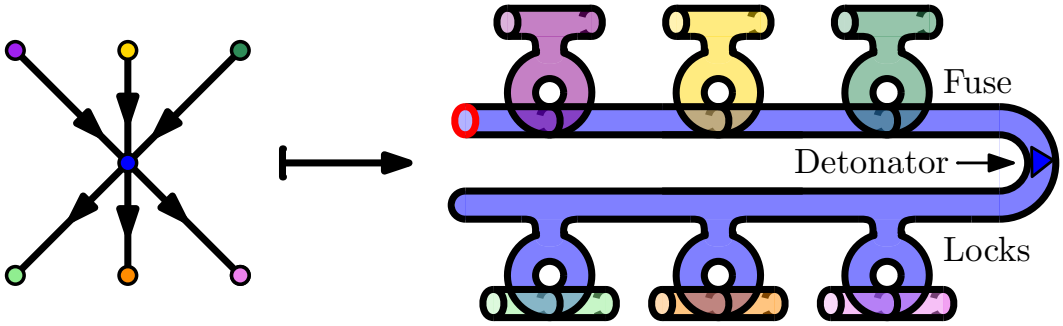


Figure 11 Schematic vertex gadget for a vertex v: the fuse $\mathcal{F}(v)$ and the lock region $\mathcal{L}(v)$.

Ouroboroi as obstructions. Fix a directed cycle $C = (v_1, \ldots, v_\ell)$ in D. For each arc (v_i, v_{i+1}) (indices modulo ℓ) we use one outgoing handle of $\mathcal{L}(v_i)$ and glue it so that it pinches the fuse $\mathcal{F}(v_{i+1})$ as described above. Taking exactly these pieces along the cycle yields a closed ring of vertex gadgets in which each lock pinches the next fuse; because of the visual similarity to a snake eating its own tail, we call such a ring an *ℓ-ouroboros*, see Figure 12. Along an ouroboros, each fuse can be collapsed from its free boundary until it reaches the first pinch, but it cannot be collapsed past that point while the corresponding lock is present. By construction, every edge in this ring belongs to at least two triangles (coming from the fuse, the lock, or their intersection), so inside this subcomplex there are no free edges and hence no elementary collapses. In particular, an ouroboros persists under any sequence of elementary collapses until at least one of its locks is destroyed, and we will use these rings as obstructions witnessing the presence of directed cycles in D.

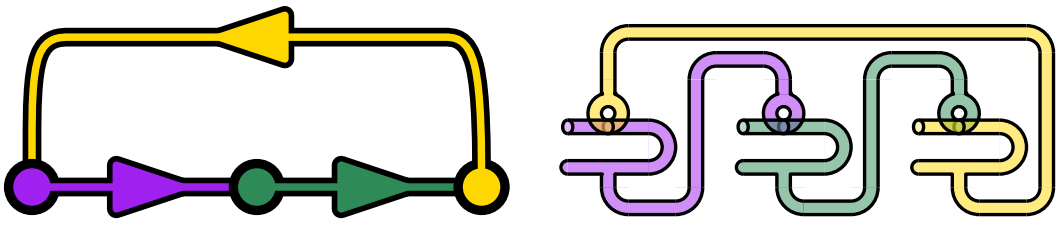


Figure 12 A directed 3-cycle in D and the corresponding 3-ouroboros: three vertex gadgets arranged in a ring, each lock pinching the next fuse. No edge on this ring is free, so it remains non-erasible until some detonator is deleted.

4.2 Correctness of the reduction

Let Y be the complex constructed from a DFVS instance D, and for each vertex v let $\mathrm{deton}(v)$ be its detonator triangle in the gadget Y^v. We show that this construction yields a parameter-preserving polynomial-time reduction from DFVS to Erasibility: for every integer s, D has a feedback vertex set of size at most s if and only if Y becomes erasible after deleting at most s triangles.

Forward direction. Let $S \subseteq V(D)$ be a feedback vertex set of size at most s, and let $T := \{\mathrm{deton}(v) \mid v \in S\}$. Deleting $\mathrm{deton}(v)$ makes the entire lock $\mathcal{L}(v)$ collapsible and removes all its pinches on neighbouring fuses, so in $Y \setminus T$ we can collapse every lock $\mathcal{L}(v)$ with $v \in S$. Since $D \setminus S$ is acyclic, we can order its vertices topologically and, in that order, collapse each remaining gadget Y^v: once all incoming locks to $\mathcal{F}(v)$ have been removed, the fuse $\mathcal{F}(v)$ collapses completely, and by the gadget design the remaining lock $\mathcal{L}(v)$ then collapses as well. At the end only gadgets for vertices in S remain, but their locks have already been removed and their fuses are just cylinders with a free boundary, so they also collapse. Thus $Y \setminus T$ is erasible after deleting $|T| = |S| \leq s$ triangles.

Backward direction. Conversely, let T be a set of at most s triangles such that $Y \setminus T$ is erasible, and define $S' := \{\, v \in V(D) \mid Y^v \cap T \neq \emptyset \,\}$. Clearly $|S'| \leq |T| \leq s$. Suppose for contradiction that $D \setminus S'$ still contains a directed cycle $C = (v_1, \ldots, v_\ell)$. By construction, the gadgets $Y^{v_1}, \ldots, Y^{v_\ell}$ contain an ℓ-ouroboros subcomplex Z. Since no gadget on C intersects T, Z is disjoint from T, and every edge of Z still lies in at least two triangles in $Y \setminus T$. Hence no edge of Z is ever free, so no sequence of elementary collapses can remove Z, and $Y \setminus T$ is not erasible – a contradiction. Thus $D \setminus S'$ is acyclic, and S' is a feedback vertex set of size at most s.

4.3 Preserving width via WiPS

Width blow-up. Hardness reductions for problems parameterized by treewidth must control the width of the target instance: a naïve "glue all gadgets at once" construction can easily turn a bounded-treewidth graph into a complex whose Hasse diagram has very large treewidth. In our setting, globally attaching all vertex gadgets Y^v would allow long fuses and many handles to interact and form large grid-like regions in the Hasse diagram, even when D itself has small treewidth; see [5] for a concrete construction or Figure 13 for intuition.

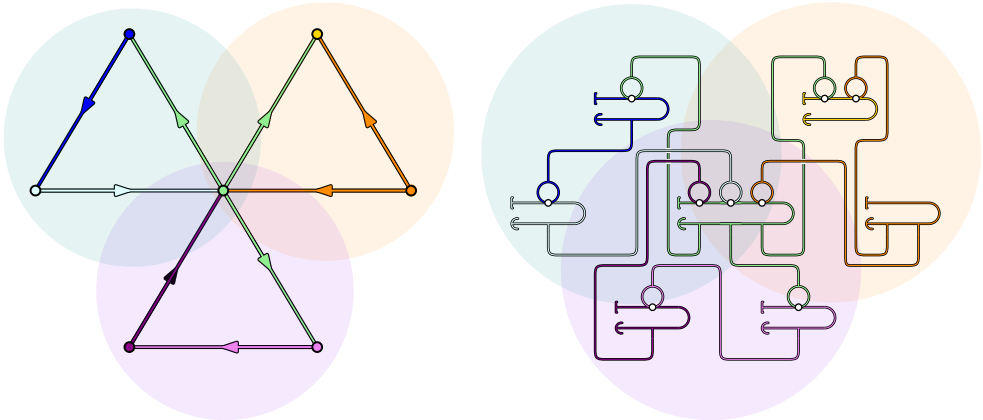


Figure 13 Naïve global gluing of all gadgets. The digraph (top) has small treewidth, but the assembled space (bottom) may develop large intertwined regions in the Hasse diagram, with no a priori width bound.

WiPS. To avoid this blow-up, we assemble the same gadgets *incrementally* along a nice tree decomposition of D, following the Width Preserving Strategy (WiPS) [43]. For each bag X_t we maintain a partial complex Y_t together with a small interface of boundary circles associated with the vertices in X_t, and when moving from a child bag to its parent we only apply constant-size local updates touching this interface: at an introduce-vertex bag we attach a fresh gadget $\mathcal{F}(v) \cup \mathcal{L}(v)$; at an introduce-edge bag we add a single pinching handle between $\mathcal{L}(u)$ and $\mathcal{F}(v)$; at a forget-vertex bag we cap off the remaining boundary components of v; and at a join bag we merge the two partial gadgets for each v along a constant-size interface (on the lock side via a pair-of-pants, on the fuse side by attaching both child cylinders to a common boundary circle and extending it by a short cylinder). Intuitively, each bag only "sees" a bounded portion of the complex, and only a bounded number of new simplices is introduced per bag; see Figure 14.

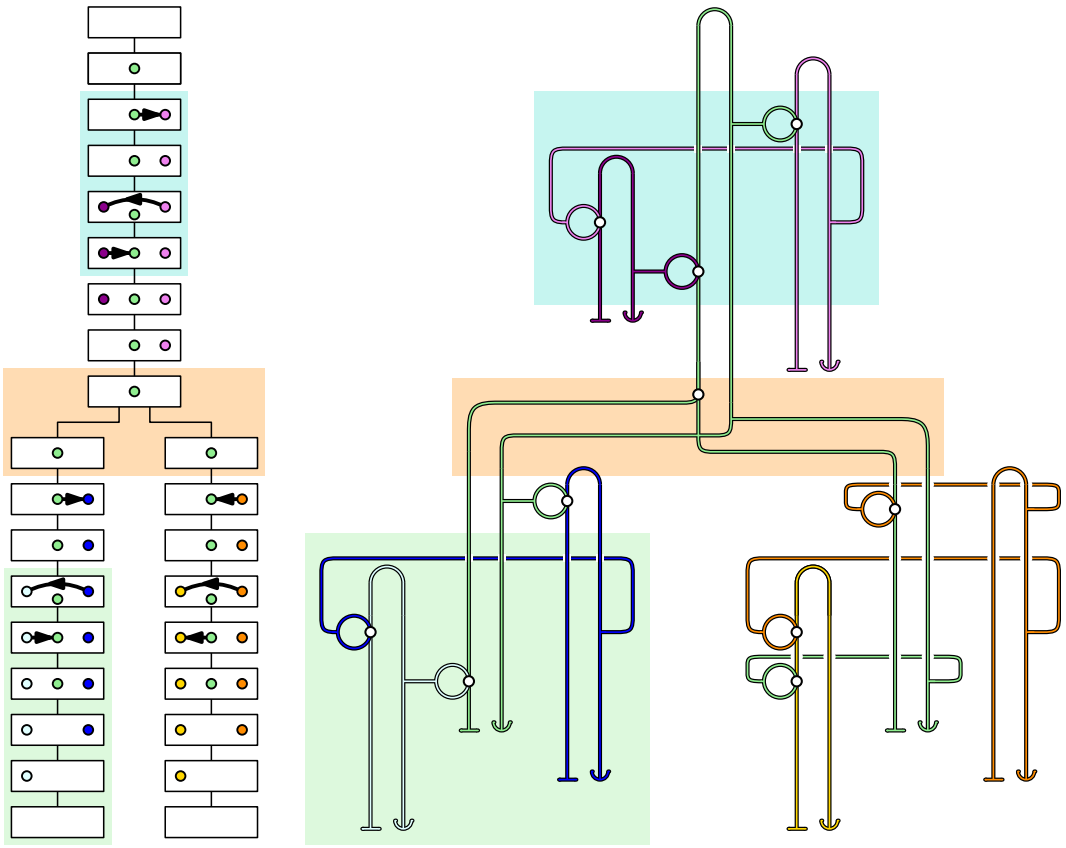


Figure 14 WiPS-style construction. Left: a nice tree decomposition of the digraph in Figure 13 Right: the space is grown bag-by-bag, each bag exposing only a small interface and adding only constant-size pieces. This ensures that the Hasse diagram maintains treewidth $O(k)$.

Summary. Instantiating WiPS with our gadgets, we obtain that if D has treewidth k, then the complex Y produced by the above construction has a Hasse diagram of treewidth at most $ck + c_0$ for fixed constants c, c_0 independent of D, and $|Y|$ remains polynomial in $|D|$. A detailed WiPS induction invariant and the exact bag-by-bag construction are in [39, App. B]. Here we only use this width bound together with the gadget behavior described above. Together with the correctness of the DFVS-to-Erasibility reduction proved above, this yields Theorem 13.

5 Discussion

Table 1 summarises our treewidth-parameterized bounds. For FMM/FMO on digraphs, OMM on Hasse diagrams, and 2D Erasability with coface degree at most 4 we obtain algorithms running in time $2^{O(k \log k)}n$, and our width-preserving reduction from DFVS shows that no $2^{o(k \log k)}n^{O(1)}$ algorithm exists under ETH. Thus these problems have optimal $2^{\Theta(k \log k)}n^{O(1)}$ dependence on treewidth. The rows with unknown ETH bounds in Table 1 show that the same dynamic programme extends to OMM on triangulated manifolds and to negative-weight FMM/FMO, and it remains open whether ETH lower bounds also carry over or whether the extra structure allows faster algorithms. Our algorithm also applies to AC-FM on bipartite graphs. In the single-level bipartite incidence-graph setting used in the matching-based formulation of discrete Morse theory (e.g. spine graphs), AC-FM, URM [26], and the corresponding discrete Morse matching problem are equivalent; see [12]. Hence we obtain a $2^{O(k \log k)}n^{O(1)}$-time algorithm for bipartite AC-FM/URM. Our ETH lower bound already holds for bipartite incidence graphs of maximum degree at most 4, in particular for those arising from 2-complexes, so the $2^{\Theta(k \log k)}n^{O(1)}$ dependence is ETH-tight in the bipartite setting. By contrast, on general graphs the recent treewidth-based dynamic programme of [15] runs in $2^{O(k^2)}n^{O(1)}$ time. Whether the $2^{O(k \log k)}$ dependence can also be achieved for general graphs remains open.

Table 1 Upper and lower running-time bounds in the treewidth parameter k for the problems considered in this paper, suppressing polynomial factors in n. The first block contains the problems we study explicitly, ordered from more general to more restricted settings. The bottom block records bounds for AC-FM/URM; the bounded-degree row indicates that the ETH lower bound already holds for bipartite incidence graphs arising from 2D Erasability.

Problem	Setting	Algorithm	ETH
FMM/FMO	General digraphs, any weights	$2^{O(k \log k)}$	$2^{o(k \log k)}$
FMM/FMO	General digraphs, negative weights	$2^{O(k \log k)}$	–
OMM	DAGs/Hasse diagrams	$2^{O(k \log k)}$	$2^{o(k \log k)}$
OMM	Triangulated d-manifolds	$2^{O(k \log k)}$	–
2D Erasability	Coface degree ≤ 4, unweighted	$2^{O(k \log k)}$	$2^{o(k \log k)}$
Conjectured $\rightarrow$	Coface degree ≤ 3, embeddable in $\mathbb{R}^3$	$2^{O(k \log k)}$	$2^{o(k \log k)}$
AC-FM/URM	General graphs, algorithm from [15]	$2^{O(k^2)}$	$2^{o(k \log k)}$
AC-FM/URM	Bipartite graphs	$2^{O(k \log k)}$	$2^{o(k \log k)}$
AC-FM/URM	Bipartite graphs, maximum degree ≤ 4	$2^{O(k \log k)}$	$2^{o(k \log k)}$

Representation matters. Forman's correspondence identifies discrete Morse matchings with Morse functions or orders on the underlying Hasse diagram, but these viewpoints behave very differently for treewidth-based algorithms. Most prior work, including Hasse-based formulations of OMM and the parameterized algorithm of [12], stays in the matching language: discrete gradients are matchings and gradient paths are alternating paths. In that setting one either appeals to meta-theorems such as Courcelle's theorem, with non-elementary state spaces, or designs ad-hoc state summaries based on alternating-path patterns on each bag of a tree decomposition. Figure 15 shows that such summaries are delicate: alternating reachability is not transitive, and merging states solely on the basis of their alternating pattern

can silently discard globally optimal extensions. In particular, our example suggests that the union–find based invariant used in [12], which implicitly treats alternating connectivity as an equivalence relation, is too coarse as stated; one really needs to know which vertices are alternating-reachable from which, not just which "components" they lie in. The recent AC-FM/URM dynamic program of [15] can be seen as a systematic matching-based repair: on each bag it stores an "alternating matrix" recording, for every pair of vertices, whether they are connected by an alternating path with respect to the current matching, that is, the full relation $R_M \subseteq B \times B$ of alternating reachability. This avoids over-pruning and, on bipartite incidence graphs (such as spines or Hasse diagrams), yields a faithful matching-based implementation of discrete Morse matchings, but at the price of a $2^{\Theta(k^2)}$ state space for bags of size $O(k)$, as reflected in Table 1. Beyond the bipartite/Hasse case the relationship between AC-FM and FMM breaks down, and we do not currently see how to compress alternating information on general graphs to obtain a $2^{O(k \log k)}$ dependence without a substantially different representation.

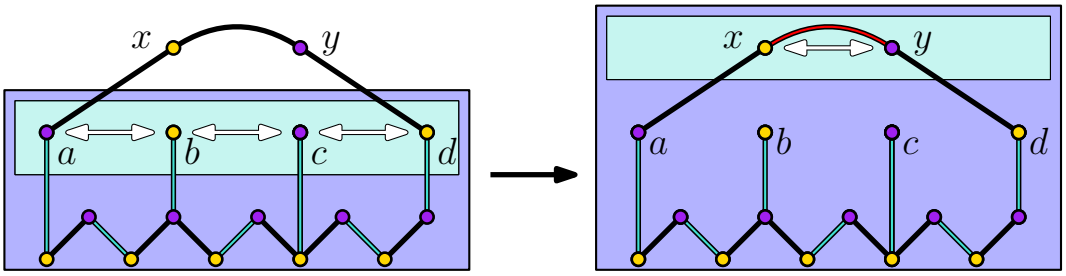


Figure 15 Summaries based only on alternating-path patterns can over-prune. **Left:** the DP merges states whenever the alternating-path pattern on the current bag is the same; in particular, states that differ only in how they treat a and d are identified. **Right:** after introducing x, y, the matching $\{ax, yd\}$ is a valid extension and can be arranged to be uniquely optimal, but no surviving state represents it, because all states that kept a and d separate were merged earlier.

Conclusion. We give an order-based dynamic program for Optimal Morse Matching on bounded-treewidth complexes with running time $2^{O(k \log k)}n$ and, via WiPS-style width-preserving reductions, ETH-tight lower bounds that persist even under strong restrictions on the input. For discrete Morse theory, our results suggest that for treewidth, taking the path of functions and orderings has made all the difference. Looking ahead, the road keeps going:

- **Three gaps suggested by Table 1.** First, prove the conjectured ETH-tight lower bound for 2D Erasability on complexes of coface degree ≤ 3 embeddable in $\mathbb{R}^3$. Second, for triangulated manifolds can we obtain a $2^{O(k)}n^{O(1)}$-time algorithm for OMM? Third, for AC-FM/URM we have an ETH-tight $2^{\Theta(k \log k)}n^{O(1)}$ running time on bipartite graphs, while on general graphs the best known DP runs in $2^{O(k^2)}n^{O(1)}$ time [15]. Can we show under ETH that a quadratic dependence on treewidth in the exponent is unavoidable in the general-graph case?
- **A locality constraint for** OMM**.** With solution-size parameterizations remaining W[P]-hard [12, 4], it is natural to look for complementary structural parameters. One concrete direction is to enforce *local* simplification by bounding (or minimizing) the length of the longest gradient path, equivalently the longest alternating path in the Hasse diagram after reversing matched edges. What is the complexity of OMM under such a locality constraint, and how does it interact with bounded treewidth?

- **WiPS beyond this paper.** Our ETH lower bound relies on the Width Preserving Strategy (WiPS) [43], which enables reductions to be carried out bag-by-bag while keeping treewidth under control. Can WiPS be developed into a general toolbox for proving tight treewidth lower bounds (and perhaps transferring XNLP-hardness) for other problems studied via treewidth in topology and geometry, for example quantum invariants and decision problems on triangulations [10, 13, 6]?

References

1 Madjid Allili, Tomasz Kaczynski, Claudia Landi, and Filippo Masoni. Acyclic partial matchings for multidimensional persistence: Algorithm and combinatorial interpretation. *Journal of Mathematical Imaging and Vision*, 61(2):174–192, 2019. doi:10.1007/s10851-018-0843-8.

2 Ulrich Bauer, Carsten Lange, and Max Wardetzky. Optimal topological simplification of discrete functions on surfaces. *Discrete & Computational Geometry*, 47(2):347–377, 2012. doi:10.1007/s00454-011-9350-z.

3 Ulrich Bauer and Abhishek Rathod. Hardness of approximation for Morse matching. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2663–2674. SIAM, 2019. doi:10.1137/1.9781611975482.165.

4 Ulrich Bauer and Abhishek Rathod. Parameterized inapproximability of Morse matching. *Computational Geometry*, 126:102148, 2025. doi:10.1016/j.comgeo.2024.102148.

5 Mitchell Black, Nello Blaser, Amir Nayyeri, and Erlend Raa Vågset. ETH-tight algorithms for finding surfaces in simplicial complexes of bounded treewidth. In *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.17.

6 Mitchell Black and Amir Nayyeri. Finding minimum bounded and homologous chains in simplicial complexes with bounded-treewidth 1-skeleton. arXiv, 2021. arXiv:2107.10339.

7 Nello Blaser, Morten Brun, Lars M. Salbu, and Erlend Raa Vågset. The parameterized complexity of finding minimum bounded chains. *Computational Geometry*, 122:102102, 2024. doi:10.1016/j.comgeo.2024.102102.

8 Nello Blaser and Erlend Raa Vågset. Homology localization through the looking-glass of parameterized complexity theory. *Journal of Applied and Computational Topology*, 9:16, 2025. doi:10.1007/s41468-025-00212-0.

9 Marthe Bonamy, Łukasz Kowalik, Jesper Nederlof, Michał Pilipczuk, Arkadiusz Socała, and Marcin Wrochna. On directed feedback vertex set parameterized by treewidth. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Graph-Theoretic Concepts in Computer Science (WG 2018)*, volume 11159 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2018. Also available as arXiv:1707.01470. doi:10.1007/978-3-030-00256-5_6.

10 Benjamin A. Burton. The HOMFLY–PT Polynomial is Fixed-Parameter Tractable. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2018.18.

11 Benjamin A. Burton and Rodney G. Downey. Courcelle's theorem for triangulations. *Journal of Combinatorial Theory, Series A*, 146:264–294, February 2017. doi:10.1016/j.jcta.2016.10.001.

12 Benjamin A. Burton, Thomas Lewiner, João Paixão, and Jonathan Spreer. Parameterized complexity of discrete Morse theory. *ACM Transactions on Mathematical Software*, 42(1):6:1–6:24, March 2016. doi:10.1145/2738034.

13 Benjamin A. Burton, Clément Maria, and Jonathan Spreer. Algorithms and complexity for Turaev–Viro invariants. *Journal of Applied and Computational Topology*, 2(1-2):33–53, 2018. doi:10.1007/s41468-018-0016-2.

14 Frédéric Cazals, Frédéric Chazal, and Thomas Lewiner. Molecular shape analysis based upon the Morse–Smale complex and the Connolly function. In *Proceedings of the 19th Annual Symposium on Computational Geometry*, pages 351–360, 2003. doi:10.1145/777792.777845.

15 Juhi Chaudhary, Ignasi Sau, and Meirav Zehavi. A parameterized perspective on uniquely restricted matchings. *Procedia Computer Science*, 273:509–516, 2025. Also available as arXiv:2508.12004. doi:10.1016/j.procs.2025.10.339.

16 Justin Curry, Robert Ghrist, and Vidit Nanda. Discrete Morse theory for computing cellular sheaf cohomology. *Foundations of Computational Mathematics*, 16(4):875–897, 2016. doi:10.1007/s10208-015-9266-8.

17 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, Cham, 2015. doi:10.1007/978-3-319-21275-3.

18 Olaf Delgado-Friedrichs, Vanessa Robins, and Adrian Sheppard. Skeletonization and partitioning of digital images using discrete Morse theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):654–666, 2015. doi:10.1109/TPAMI.2014.2346172.

19 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, New York, NY, 1999. doi:10.1007/978-1-4612-0515-9.

20 Michael Farber. *Invitation to Topological Robotics*. Zurich Lectures in Advanced Mathematics. European Mathematical Society (EMS), Zürich, 2008. doi:10.4171/054.

21 Robin Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134(1):90–145, 1998. doi:10.1006/aima.1997.1650.

22 Robin Forman. A user's guide to discrete Morse theory. *Séminaire Lotharingien de Combinatoire*, 48:B48c, 2002. URL: https://www.mat.univie.ac.at/~slc/wpapers/s48forman.html.

23 Ulderico Fugacci, Federico Iuricich, and Leila De Floriani. Computing discrete Morse complexes from simplicial complexes. *Graphical Models*, 103:101023, 2019. doi:10.1016/j.gmod.2019.101023.

24 Robert Ghrist. Configuration spaces, braids, and robotics. In *Braids: Introductory Lectures on Braids, Configurations and Their Applications*, pages 263–304. World Scientific, 2010. doi:10.1142/9789814291415_0004.

25 Xavier Goaoc, Pavel Paták, Zuzana Patáková, Martin Tancer, and Uli Wagner. Shellability is NP-complete. *Journal of the ACM*, 66(3):21, 2019. doi:10.1145/3314024.

26 Martin Charles Golumbic, Tom Hirst, and Michael Lewenstein. Uniquely restricted matchings. *Algorithmica*, 31(2):139–154, 2001. doi:10.1007/s00453-001-0004-z.

27 Shaun Harker, Konstantin Mischaikow, Marian Mrozek, and Vidit Nanda. Discrete Morse theoretic algorithms for computing homology of complexes and maps. *Foundations of Computational Mathematics*, 14(1):151–184, 2014. doi:10.1007/s10208-013-9145-0.

28 Kristóf Huszár and Jonathan Spreer. 3-Manifold Triangulations with Small Treewidth. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44:1–44:20, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2019.44.

29 Kristóf Huszár and Jonathan Spreer. On the width of complicated JSJ decompositions. *Discrete & Computational Geometry*, 74(4):917–943, 2025. doi:10.1007/s00454-025-00746-1.

30 Kristóf Huszár, Jonathan Spreer, and Uli Wagner. On the treewidth of triangulated 3-manifolds. *Journal of Computational Geometry*, 10(2):70–98, 2019. doi:10.20382/jocg.v10i2a5.

31 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

32 Michael Joswig and Marc E. Pfetsch. Computing optimal Morse matchings. *SIAM Journal on Discrete Mathematics*, 20(1):11–25, 2006. doi:10.1137/S0895480104445885.

33 Harish Kannan, Emil Saucan, Indrava Roy, and Areejit Samal. Persistent homology of unweighted complex networks via discrete Morse theory. *Scientific Reports*, 9:13817, 2019. doi:10.1038/s41598-019-50202-3.

34 Thomas Lewiner, Helio Lopes, and Geovan Tavares. Toward optimality in discrete Morse theory. *Experimental Mathematics*, 12(3):271–285, 2003. doi:10.1080/10586458.2003.10504498.

35 Thomas Lewiner, Helio Lopes, and Geovan Tavares. Applications of Forman's discrete Morse theory to topology visualization and mesh compression. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):499–508, 2004. doi:10.1109/TVCG.2004.18.

36 Rémy Malgouyres and Angel R. Francés. Determining whether a simplicial 3-complex collapses to a 1-complex is NP-complete. In *Discrete Geometry for Computer Imagery (DGCI 2008)*, volume 4992 of *Lecture Notes in Computer Science*, pages 177–188. Springer, 2008. doi:10.1007/978-3-540-79126-3_17.

37 Marston Morse. *The Calculus of Variations in the Large*, volume 18 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, RI, 1996. Reprint of the 1934 original. doi:10.1090/coll/018.

38 Soham Mukherjee. Denoising with discrete Morse theory. *The Visual Computer*, 37(9):2883–2894, 2021. doi:10.1007/s00371-021-02255-7.

39 Geevarghese Philip and Erlend Raa Vågset. ETH-tight complexity of optimal Morse matching on bounded-treewidth complexes, 2026. Full version. arXiv:2603.05406.

40 Abhishek Rathod, Talha Bin Masood, and Vijay Natarajan. Approximation algorithms for Max Morse Matching. *Computational Geometry*, 61:1–23, 2017. doi:10.1016/j.comgeo.2016.10.002.

41 Nicholas A. Scoville. *Discrete Morse Theory*, volume 90 of *Student Mathematical Library*. American Mathematical Society, Providence, RI, 2019. doi:10.1090/stml/090.

42 Martin Tancer. Recognition of collapsible complexes is NP-complete. *Discrete & Computational Geometry*, 55(1):21–38, 2016. doi:10.1007/s00454-015-9747-1.

43 Erlend Raa Vågset. *Optimal parameterized algorithms for solving NP-hard problems in topology*. PhD thesis, University of Bergen, Bergen, Norway, 2024. URL: https://urn.nb.no/URN:NBN:no-nb_pliktmonografi_000039725.

Expansion of Trivariate Polynomials Using Proximity

Orit E. Raz
Department of Mathematics, Ben-Gurion University of the Negev, Be'er Sheva, Israel

Abstract

We extend the proximity technique of Solymosi and Zahl [9] to the setting of trivariate polynomials. In particular, we prove the following result: Let $f(x,y,z) = (x-y)^2+(\varphi(x)-z)^2$, where $\varphi(x) \in \mathbb{R}[x]$ has degree at least 3. Then, for every finite $A, B, C \subset \mathbb{R}$ each of size n, one has $|f(A,B,C)| = \Omega(n^{5/3-\varepsilon})$, for every $\varepsilon > 0$, where the constant of proportionality depends on ε and on $\deg(\varphi)$. This improves the previous exponent 3/2, due to Raz, Sharir, and De Zeeuw [7]. To the best of our knowledge, prior to this work no trivariate polynomial was known to have expansion exceeding $\Omega(n^{3/2})$.

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems

Keywords and phrases Polynomial Expansion, Elekes–Rónyai theorem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.86

1 Introduction

In many cases in combinatorial geometry, counting questions involving distances, slopes, collinearity, etc., can be reformulated as analogous counting questions involving grid points lying on certain algebraic varieties. A unified study of such problems began with a question of Elekes [1] about expansion of bivariate real polynomials $f(x,y)$. Specifically, he asked: For a bivariate polynomial $f \in \mathbb{R}[x,y]$ and given finite sets $A, B \subset \mathbb{R}$, how small can be the image set

$$f(A,B) = \{f(a,b) \mid a \in A, b \in B\}.$$

Elekes conjectured that the image of f on an $n \times n$ Cartesian product must be of cardinality superlinear in n, unless f has a very concrete *special form*. This was confirmed in 2000 by Elekes and Rónyai [2], who proved the following dichotomy: Either f is one of the forms

$$\begin{aligned} &f(x,y) = h(p(x)+q(y)) \quad \text{or} \\ &f(x,y) = h(p(x)q(y)), \end{aligned} \tag{1}$$

for some univariate real polynomials p, q, h, or, otherwise, for every finite $A, B \subset \mathbb{R}$, each of size n, we have

$$|f(A,B)| = \omega(n). \tag{2}$$

In [8], Raz, Sharir and Solymosi introduced a new proof of the Elekes–Rónyai theorem, which also yields improved bounds on the expansion of f. Roughly speaking, for a given $f \in \mathbb{R}[x,y]$ and two finite sets $A, B \subset \mathbb{R}$, they bounded the number of quadruples $((a,b),(a',b')) \in (A \times B)^2$ satisfying $f(a,b) = f(a',b')$ by reducing it to a point-curve incidence problem in the plane. Concretely, incidences between the point set $A \times A$ and the family of curves $\{\gamma_{b,b'} \mid (b,b') \in B \times B\}$, where $\gamma_{b,b'}$ is given by the equation

$$f(x,b) = f(y,b').$$

If these curves are distinct, known incidence bounds can be applied to obtain the desired estimate. However, it may happen that many of the curves coincide, in which case the

© Orit E. Raz;
licensed under Creative Commons License CC-BY 4.0

incidence bound breaks down. Raz, Sharir, and Solymosi showed that this "failure" occurs if and only if f has a special form, namely one of the forms described in (1). In the non-special case, their analysis yields the lower bound $\Omega(n^{4/3})$

Solymosi and Zahl [9] recently improved the expansion bound for f in the non-special case, establishing the lower bound $\Omega(n^{3/2})$. Their argument builds on the framework of [8], but applies the incidence bound to carefully selected subsets of points and curves. In particular, they restrict to points $(a, a') \in A \times A$ with a and a' sufficiently close, and, similarly, to curves $\gamma_{b,b'}$ with parameters $(b, b') \in B \times B$ with b and b' sufficiently close. This refinement, referred to as the *proximity method*, overcomes a loss incurred in the earlier argument from an application of the Cauchy–Schwarz inequality.

A 3-variate analogue of the Elekes–Rónyai theorem was studied in Raz, Sharir, and De Zeeuw [7]. The case of $k \geq 4$ variables is studied by Raz and Shem Tov in [4].

▶ **Theorem 1** ([7, 4]). *Let $k \geq 3$ and assume that $f \in \mathbb{R}[x_1, \ldots, x_k]$ depends non-trivially on each of its varaiables. Then one of the following holds:*

(i) *For every finite $A_1, \ldots, A_k \subset \mathbb{R}$, with $|A_i| = n$, for $i = 1, \ldots, k$, one has*

$$|f(A_1, \ldots, A_k)| = \Omega(n^{3/2}),$$

where the constant of proportionality depends only on $\deg(f)$ and on k.

(ii) *f is of one of the special forms*

$$f(x_1, \ldots, x_k) = h(p_1(x_1) + \cdots + p_k(x_k)) \tag{3}$$
$$f(x_1, \ldots, x_k) = h(p_1(x_1) \cdot \ldots \cdot p_k(x_k))$$

for some univariate real polynomials $p_1, \ldots, p_k, h$.

Similar to the bivariate case, the analysis for $k = 3$ also reduces to a point-curve incidence problem in the plane. Specifically, for a trivariate polynomial $f \in \mathbb{R}[x, y, z]$ and finite sets $A, B, C \subset \mathbb{R}$, define $D := f(A, B, C)$ and consider incidences between the point set $A \times D$ and the family of curves $\{\gamma_{b,c} \mid (b, c) \in B \times C\}$, where $\gamma_{b,c}$ is given by the equation

$$w = f(x, b, c).$$

In some respects, the trivariate case is simpler than the bivariate one. Here, the structure of f is more directly reflected in properties of the variety $w = f(x, y, z)$, whereas in the bivariate case one must analyze the more intricate variety $f(x, y) = f(z, w)$. Moreover, in the trivariate setting the argument avoids an application of the Cauchy–Schwarz inequality, therefore eliminating the loss in the estimate that arises in the bivariate case.

As mentioned above, the proximity method of Solymosi and Zahl [9] improves the bivariate bound precisely by overcoming this loss from the Cauchy–Schwarz step. For this reason, it is not immediately clear how their method could extend to the trivariate case.

In this paper, we show how proximity can in fact be used to obtain a stronger expansion bound for trivariate polynomials. In particular, we establish an improved bound for a concrete family of trivariate polynomials. Namely, we prove the following main result.

▶ **Theorem 2.** *Let*

$$f(x, y, z) = (x - y)^2 + (\varphi(x) - z)^2,$$

where $\varphi(x)$ is a univariate real polynomial of degree at least 3. Then, for every $\varepsilon > 0$ and any finite sets $A, B, C \subset \mathbb{R}$, each of size n, we have

$$|f(A, B, C)| = \Omega(n^{5/3-\varepsilon}),$$

where the constant of proportionality depends on $\deg \varphi$ and on ε.

This improves upon the previous bound of $\Omega(n^{3/2})$, which follows from Theorem 1. To the best of our knowledge, prior to this work no trivariate polynomial was known to have expansion exceeding $\Omega(n^{3/2})$.

We remark that our approach is more general and extends to other trivariate polynomials. A natural direction for future research is to determine the precise subfamily of polynomials $\mathcal{F} \subset \mathbb{R}[x,y,z]$ for which the method yields the improved lower bound $\Omega(n^{5/3-\varepsilon})$.

2 Preliminaries

2.1 Point-curve incidences in the plane

For a finite set of points $\mathcal{P} \subset \mathbb{R}^2$ and a finite set of planar curves $\mathcal{C}$, we let $I(\mathcal{P},\mathcal{C})$ denote the set of point-curve incidences; that is

$$I(\mathcal{P},\mathcal{C}) = \{(p,\gamma) \in \mathcal{P} \times \mathcal{C} \mid p \in \gamma\}.$$

The classical Szemerédi–Trotter theorem [10] asserts that, for the special case where $\mathcal{C}$ is a set of lines, and putting $m := |\mathcal{P}|$ and $n := |\mathcal{C}|$, one has

$$|I(\mathcal{P},\mathcal{C})| = O\left(m^{2/3}n^{2/3} + m + n\right).$$

Since the Szemerédi–Trotter paper, numerous alternative proofs and related problems have been studied. For our result, we require an extension of Szemerédi–Trotter to point-curve incidence problems, established by Sharir and Zahl [6], in which the curves are algebraic and form an s-dimensional family. We now recall the relevant definitions from their work.

A bivariate polynomial $h \in \mathbb{R}[x,y]$ of degree at most D is a linear combination of the form $h(x,y) = \sum_{0\le i+j\le D} c_{ij}x^iy^j$. Note that the number of monomials x^iy^j such that $0 \le i+j \le D$ is $\binom{D+2}{2}$. In this sense, every point $\vec{c} \in \mathbb{R}^{\binom{D+2}{2}}$ (other than the all-zero vector) can be associated with a curve in $\mathbb{R}^2$, given by the zeroset of the bivariate polynomial whose coefficients are the entries of $\vec{c}$. If $\lambda \neq 0$, then f and λf have the same zero-set. Thus, the set of algebraic curves that can be defined by a polynomial of degree at most D in $\mathbb{R}^2$ can be identified with the points in the projective space $\mathbf{P}\mathbb{R}^{\binom{D+2}{2}}$.

Define an *s-dimensional family of plane curves of degree at most D* to be an algebraic variety $F \subset \mathbf{P}R^{\binom{D+2}{2}}$ such that $\dim(F) = s$. We will call the degree of the variety F the *complexity of the family*.

They then proved the following incidence bound:

▶ **Theorem 3** (Sharir–Zahl [6]). *Let Γ be a set of n algebraic plane curves that belong to an s-dimensional family of curves of degree at most D of constant complexity at most K, no two of which share a common irreducible component. Let P be a set of m points in the plane. Then for any $\varepsilon > 0$, the number of incidences $|I(P,\Gamma)|$ between the points of P and the curves of Γ satisfies*

$$|I(P,\Gamma)| = O_\varepsilon\left(m^{\frac{2s}{5s-4}}n^{\frac{5s-6}{5s-4}+\varepsilon}\right) + O\left(m^{2/3}n^{2/3} + m + n\right),$$

where the constant of proportionality depends on s, K, D, and in the first term also on ε.

2.2 Symmetry of curves

Given a set $S \subset \mathbb{R}^2$ and a transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$, we say that T *fixes* S if $T(S) = S$. We say that a transformation T is a *symmetry* of a plane algebraic curve C if T is an isometry of $\mathbb{R}^2$ and fixes C. Recall that an isometry of $\mathbb{R}^2$ is either a rotation, a translation, or a glide reflection (a reflection combined with a translation).

The following lemma is proved by Pach and De Zeeuw [3, Lemma 2.5].

▶ **Lemma 4** (Pach and De Zeeuw [3, Lemma 2.5])**.** *An irreducible plane algebraic curve of degree d has at most $4d$ symmetries, unless it is a line or a circle.*

2.3 Complete bipartite graphs

We introduce some notation and recall some properties in the spirit of Raz–Solymosi [5, Section 3]. For completeness we give all the details here. Let $\mathbf{p} = (p_1, \ldots, p_k), \mathbf{p}' = (p'_1, \ldots, p'_k) \in \mathbb{R}^2 \times \cdots \times \mathbb{R}^2 \cong \mathbb{R}^{2k}$ be two k-tuples of points in $\mathbb{R}^2$. Define

$$\Sigma_{\mathbf{p},\mathbf{p}'} = \{(q, q') \in \mathbb{R}^2 \times \mathbb{R}^2 \mid \|p_i - q\| = \|p'_i - q'\| \text{ for each } i = 1, \ldots, k\} \subset \mathbb{R}^2 \times \mathbb{R}^2.$$

Let $\sigma_{\mathbf{p},\mathbf{p}'}$ (respectively, $\sigma'_{\mathbf{p},\mathbf{p}'}$) denote the projection of $\Sigma_{\mathbf{p},\mathbf{p}'}$ to the first (respectively, last) copy of $\mathbb{R}^2$ in $\mathbb{R}^2 \times \mathbb{R}^2$.

▶ **Lemma 5.** *Let $\mathbf{p} = (p_1, p_2, p_3), \mathbf{p}' = (p'_1, p'_2, p'_3) \in (\mathbb{R}^2)^3$, and assume that p_1, p_2, p_3 are pairwise distinct. Then either*
(i) $\mathbf{p}$ and $\mathbf{p}'$ are congruent, or
(ii) $\Sigma_{\mathbf{p},\mathbf{p}'}$ is at most one-dimensional, and each of $\sigma_{\mathbf{p},\mathbf{p}'}$ and $\sigma'_{\mathbf{p},\mathbf{p}'}$ is contained in an algebraic curve of degree at most two.

Proof. By definition, for $(q, q') \in \Sigma_{\mathbf{p},\mathbf{p}'}$ we have

$$\|p_i - q\|^2 = \|p'_i - q'\|^2, \quad i = 1, 2, 3$$

or

$$\|p_i\|^2 - 2p_i \cdot q + \|q\|^2 = \|p'_i\|^2 - 2p'_i \cdot q' + \|q'\|^2, \quad i = 1, 2, 3.$$

Subtracting the 3rd equation from each of the first two equations, we get the system

$$\begin{aligned} \|p_1\|^2 - \|p_3\|^2 - 2(p_1 - p_3) \cdot q &= \|p'_1\|^2 - \|p'_3\|^2 - 2(p'_1 - p'_3) \cdot q' \\ \|p_2\|^2 - \|p_3\|^2 - 2(p_2 - p_3) \cdot q &= \|p'_2\|^2 - \|p'_3\|^2 - 2(p'_2 - p'_3) \cdot q' \\ \|p_3\|^2 - 2p_3 \cdot q + \|q\|^2 &= \|p'_3\|^2 - 2p'_3 \cdot q' + \|q'\|^2. \end{aligned} \tag{4}$$

The system can be rewritten as

$$\begin{aligned} \tfrac{1}{2}u - Aq &= \tfrac{1}{2}v - Bq', \\ \|p_3\|^2 - 2p_3 \cdot q + \|q\|^2 &= \|p'_3\|^2 - 2p'_3 \cdot q' + \|q'\|^2, \end{aligned}$$

where A (resp., B) is a 2×2 matrix whose ith row equals $p_i - p_3$ (resp., $p'_i - p'_3$), for $i = 1, 2$, and

$$u = \begin{pmatrix} \|p_1\|^2 - \|p_3\|^2 \\ \|p_2\|^2 - \|p_3\|^2 \end{pmatrix}, \quad v = \begin{pmatrix} \|p'_1\|^2 - \|p'_3\|^2 \\ \|p'_2\|^2 - \|p'_3\|^2 \end{pmatrix}$$

are vectors in $\mathbb{R}^2$.

Assume first that p'_1, p'_2, p'_3 are not collinear. So the matrix B is invertible and we have

$$q' = B^{-1}Aq + w,$$

for $w = \frac{1}{2}B^{-1}(v - u) \in \mathbb{R}^2$. Let $T(q) := B^{-1}Aq + w$. So $(q, q') \in \Sigma_{\mathbf{p},\mathbf{p}'}$ if and only if $q' = T(q)$. Plugging this in the 3rd equation in (4) we get

$$\|p_3\|^2 - 2p_3 \cdot q + \|q\|^2 = \|p'_3\|^2 - 2p'_3 \cdot T(q) + \|T(q)\|^2, \tag{5}$$

which defines $\sigma_{\mathbf{p},\mathbf{p}'}$. Note that this gives a conic section unless (5) is trivial, i.e., the zero equation. Note that for this to happen the quadratic part has to be zero. That is,

$$\langle q, q\rangle = \langle B^{-1}Aq, B^{-1}Aq\rangle$$

or

$$\langle q, \left(I - (B^{-1}A)^{tr}B^{-1}A\right) q\rangle = 0$$

This defines a trivial quadratic equation if and only if

$$I - (B^{-1}A)^{tr}B^{-1}A = 0$$

or

$$(B^{-1}A)^{tr} = (B^{-1}A)^{-1},$$

which means that T is an isometry of $\mathbb{R}^2$. This implies that $\mathbf{p}$ and $\mathbf{p}'$ are congruent. We conclude that $\sigma_{\mathbf{p},\mathbf{p}'}$ is either conic section given by (5) or $\mathbf{p}$ and $\mathbf{p}'$ are congruent, which completes the proof for the case that p'_1, p'_2, p'_3 are non-collinear.

By symmetry, same analysis applies also when p_1, p_2, p_3 are non-collinear. So we need to prove the lemma for the case that each of the triples p_1, p_2, p_3 and p'_1, p'_2, p'_3 is collinear. In this case we may assume without loss of generality that $p_3 = p'_3 = (0, 0)$, and $p_1 = (a, 0)$, $p'_1 = (a', 0)$, $p_2 = (b, 0)$, and $p'_2 = (b', 0)$. Note that our assumption that p_1, p_2, p_3 are pairwise distinct implies $b \neq a \neq 0$. Writing $q = (x, y)$ and $q' = (x', y')$, the system of equations defining $\Sigma_{\mathbf{p},\mathbf{p}'}$ in this case becomes

$$\begin{aligned}(x-a)^2 + y^2 &= (x'-a')^2 + (y')^2\\ (x-b)^2 + y^2 &= (x'-b')^2 + (y')^2\\ x^2 + y^2 &= (x')^2 + (y')^2\end{aligned}$$

or

$$\begin{aligned}-2ax + a^2 &= -2a'x' + (a')^2\\ -2bx + b^2 &= -2b'x' + (b')^2\\ x^2 + y^2 &= (x')^2 + (y')^2.\end{aligned}$$

Recalling our assumption that p_1, p_2, p_3 are distinct, we have $b \neq a \neq 0$. Thus the first two equations give

$$\begin{aligned}x &= (a'/a)x' - (a')^2/(2a) + a/2\\ x &= (b'/b)x' - (b')^2/(2b) + b/2.\end{aligned}$$

So we either get unique values for x, x' or otherwise $a'/a = b'/b$. In the former case both $\sigma_{\mathbf{p},\mathbf{p}'}$ and $\sigma'_{\mathbf{p},\mathbf{p}'}$ are lines parallel to the y-axis, and so in particular (ii) holds. In the latter case, write $a'/a = b'/b = t$. Assume first that $t \neq 0$. Then

$$\begin{aligned}x &= tx' + (a/2)(1-t^2)\\ x &= tx' + (b/2)(1-t^2),\end{aligned}$$

and so either $t = \pm 1$, in which case $\mathbf{p}$ and $\mathbf{p}'$ are congruent, or $a = b$ which is a contradiction to our assumption that p_1, p_2, p_3 are distinct. So we may assume that $t = 0$, which means

that $a' = b' = 0$. Assume $(q, q') \in \Sigma_{\mathbf{p},\mathbf{p}'}$ and write $q = (x, y)$ and $q' = (x', y')$. Let $r := (x')^2 + (y')^2$. So $q = (x, y)$ must satisfy the system

$$\begin{aligned}(x-a)^2 + y^2 &= r\\ (x-b)^2 + y^2 &= r\\ x^2 + y^2 &= r.\end{aligned}$$

However, recalling that $b \neq a \neq 0$, this system has no solutions. So $\Sigma_{\mathbf{p},\mathbf{p}'}$ is empty in this case. This completes the proof of the lemma. ◀

3 Proof of Theorem 2

Let f be as in the statement and let $A, B, C \subset \mathbb{R}$ be finite sets, each of size n. Set

$$D := f(A, B, C).$$

Our goal is to lower bound $|D|$.

Let

$$t = n^{3/2}/(s|D|^{1/2}),$$

where $s > 0$ is a sufficiently large constant, to be determined later. Consider the partition of each A, B, C into t consecutive segments, each containing at most $\lceil n/t \rceil$ elements. Let $\{A_i\}_{i=1}^t$, $\{B_i\}_{i=1}^t$, $\{C_i\}_{i=1}^t$ stand for the corresponding partitions. We write $a \sim a'$ if $a \neq a'$ and there exists some index $i \in [t]$ such that $a, a' \in A_i$. We write similarly $b \sim b'$ and $c \sim c'$ according to the partitions of B and C respectively. Define

$$Q := \{((a,b,c),(a',b',c')) \in (A \times B \times C)^2 \mid f(a,b,c) = f(a',b',c'),\ \ a \sim a',\ b \sim b',\ c \sim c'\}$$

We prove a lower bound on $|Q|$.

▶ **Proposition 6.** *We have*

$$|Q| = \Omega(sn^3), \tag{6}$$

where the constant of proportionality depends only on $\deg(\varphi)$.

Proof. For each $d \in D$ let

$$G_d := \{(a,b,c) \in A \times B \times C \mid f(a,b,c) = d\}.$$

This defines a partition $A \times B \times C = \bigcup_{d \in D} G_d$, and thus in particular

$$\sum_{d \in D} |G_d| = n^3.$$

Let

$$D' := \{d \in D \mid |G_d| \geq n^3/(10|D|)\}.$$

Note that

$$\begin{aligned} n^3 &= \sum_{d \in D} |G_d| \\ &= \sum_{d \in D'} |G_d| + \sum_{d \in D \setminus D'} |G_d| \\ &\leq \sum_{d \in D'} |G_d| + n^3/(10|D|) \cdot |D| \end{aligned}$$

and so

$$\sum_{d\in D'} |G_d| \geq (9/10)n^3. \tag{7}$$

Next, fix $d \in D'$. Note that for each $(i,j,k) \in [t]^3$, the set $A_i \times B_j \times C_k$ is contained in an axis-parallel box in $\mathbb{R}^3$. Moreover, the boundaries of those boxes lie in the union of $3t$ planes. Let $T_d \subset [t]^3$ denote the subset of 3-tuples of indices of boxes that intersect the surface $f(x,y,z) = d$. Note that $|T_d| = O(t^2)$, with constant of proportionality that depends only on $\deg(\varphi)$. Let $T'_d \subset T_d$ be those 3-tuples of indices for which $|G_d \cap (A_i \times B_j \times C_k)| \geq s$.

We have

$$\begin{aligned}
|G_d| &= \sum_{(i,j,k)\in T_d} |G_d \cap (A_i \times B_j \times C_k)| \\
&= \sum_{(i,j,k)\in T'_d} |G_d \cap (A_i \times B_j \times C_k)| + \sum_{(i,j,k)\in T_d \setminus T'_d} |G_d \cap (A_i \times B_j \times C_k)| \\
&\leq \sum_{(i,j,k)\in T'_d} |G_d \cap (A_i \times B_j \times C_k)| + s \cdot O(t^2) \\
&\leq \sum_{(i,j,k)\in T'_d} |G_d \cap (A_i \times B_j \times C_k)| + O(n^3/(s|D|)) \\
&\leq \sum_{(i,j,k)\in T'_d} |G_d \cap (A_i \times B_j \times C_k)| + |G_d|/2,
\end{aligned}$$

where the inequality on the fourth line is by our choice of t, and the inequality on the last line is because $d \in D'$ and assuming $s > 0$ is taken sufficiently large. Thus

$$\sum_{(i,j,k)\in T'_d} |G_d \cap (A_i \times B_j \times C_k)| \geq |G_d|/2. \tag{8}$$

Using $\binom{m}{2} \geq cm$ for $m \geq 2c+1$, we conclude that, for every $d \in D'$, one has

$$\begin{aligned}
\sum_{(i,j,k)\in T'_d} \left| \binom{G_d \cap (A_i \times B_j \times C_k)}{2} \right| &\geq \frac{s-1}{2} \sum_{(i,j,k)\in T'_d} |G_d \cap (A_i \times B_j \times C_k)| \\
&\geq \frac{s-1}{4}|G_d| \\
&\geq \frac{s}{5}|G_d|,
\end{aligned}$$

where the last inequality assumes $s \geq 5$. Thus we get

$$\begin{aligned}
|Q| &= \sum_{d\in D} \sum_{(i,j,k)\in[t]^3} \left| \binom{G_d \cap (A_i \times B_j \times C_k)}{2} \right| \\
&\geq \sum_{d\in D'} \sum_{(i,j,k)\in T'_d} \left| \binom{G_d \cap (A_i \times B_j \times C_k)}{2} \right| \\
&\geq \frac{s}{5} \sum_{d\in D'} |G_d| \\
&\geq (9/50)sn^3,
\end{aligned}$$

where the last inequality uses (7). This completes the proof of Proposition 6. ◀

Next we prove an upper bound on $|Q|$.

▶ **Proposition 7.** *For every $\varepsilon > 0$, we have*

$$|Q| \leq O_\varepsilon\left((s^2 n|D|)^{9/8+\varepsilon}\right) + 4\deg(\varphi)n^3.$$

Proof. We reduce the problem of upper bounding Q to a point-curve incidence problem in the plane. For this, we associate with each $((b,c),(b',c')) \in (B\times C)^2$ the planar curve (with coordinates (x,x')), denoted by $\gamma_{b,c,b',c'}$, given by the equation

$$f(x,b,c) = f(x',b',c').$$

Note that distinct choices of $((b,c),(b',c'))$ might give rise to the same curve, or to distinct curves that share an irreducible component. Let

$$\Gamma := \{\gamma \subset \gamma_{b,c,b',c'} \text{ is irreducible} \mid ((b,c),(b',c')) \in (B\times C)^2,\ b\sim b',\ c\sim c'\}.$$

Consider the set of parameters associated with the curves, that is, the set

$$\hat{\Gamma} := \{((b,c),(b',c')) \in (B\times C)^2 \mid\ b\sim b',\ c\sim c'\}.$$

Let $\Gamma_0 \subset \Gamma$ be the set of isolated points in Γ. We first observe that

$$|\{((a_0,b,c),(a_0',b',c')) \in Q \mid (a_0,a_0') \text{ isolated point of } \gamma_{b,c,b',c'} \text{ for } ((b,c),(b',c')) \in \hat{\Gamma}\}| = O(|\hat{\Gamma}|). \quad (9)$$

Indeed, for every $((b,c),(b',c')) \in \hat{\Gamma}$, the curve $\gamma_{b,c,b',c'}$ has $O(1)$ isolated points, each giving rise to exactly one element of Q. Thus (9) follows.

We are now interested in one-dimensional components in Γ. For each $\gamma \in \Gamma$ we define

$$m(\gamma) = \{((b,c),(b',c')) \in \hat{\Gamma} \mid \gamma \subset \gamma_{b,c,b',c'}\}$$

We show that there exists a finite subset $\Gamma_1 \subset \Gamma\setminus\Gamma_0$ such that for every $\gamma \in \Gamma\setminus(\Gamma_1\cup\Gamma_0)$ we have $|m(\gamma)| \leq 4$. Indeed, let $\gamma \in \Gamma\setminus\Gamma_0$ and suppose that $|m(\gamma)| \geq 5$. So there are three pairs $p_i := (b_i,c_i)$, $p_i' := (b_i',c_i')$, with $(p_i,p_i') \in m(\gamma)$ for $i = 1,2,3$, such that without loss of generality p_1,p_2,p_3 are distinct.

Let $\mathbf{p} = (p_1,p_2,p_3)$ and $\mathbf{p'} = (p_1',p_2',p_3')$ and recall the definition of $\Sigma_{\mathbf{p},\mathbf{p'}}$ from Section 2.3. By definition of $m(\gamma)$, this means that for every $(x,x') \in \gamma$ one has

$$((x,\varphi(x)),(x',\varphi(x'))) \in \Sigma_{\mathbf{p},\mathbf{p'}}.$$

By Lemma 5, either $\mathbf{p},\mathbf{p'}$ are congruent, or $\sigma_{\mathbf{p},\mathbf{p'}}$ is a conic section. In the latter case this implies that $\{(t,\varphi(t)) \mid t\in\mathbb{R}\}$ is a conic section, which is a contradiction because $\deg(\varphi) \geq 3$ and this curve is irreducible. Thus necessarily $\mathbf{p}$ and $\mathbf{p'}$ are congruent. Thus, there exists an isometry of $\mathbb{R}^2$, R, such that

$$(x',\varphi(x')) = R(x,\varphi(x)),\ \text{for every } (x,x') \in \gamma.$$

Since γ is infinite, this implies that R maps $(t,\varphi(t))$ to itself. By Lemma 4, we have that R is one of at most $4\deg(\varphi)$ possible isometries, which in turn determines the corresponding curve γ up to at most $4\deg(\varphi)$ possibilities. We conclude that

$$|\Gamma_1| \leq 4\deg(\varphi)$$

and

$$\text{for every } \gamma \in \Gamma\setminus(\Gamma_1\cup\Gamma_0) \text{ we have } |m(\gamma)| \leq 4. \quad (10)$$

Next, let

$$\hat{\Gamma}_1 := \{((b,c),(b',c')) \in \hat{\Gamma} \mid \text{there exists } \gamma \in \Gamma_1 \setminus \Gamma_0 \text{ such that } \gamma \subset \gamma_{b,c,b',c'}\}.$$

We claim that

$$|\{((a,b,c),(a',b',c')) \in Q \mid ((b,c),(b',c')) \in \hat{\Gamma}_1\}| \leq 4\deg(\varphi)n^3. \tag{11}$$

Indeed, by what has just been argued, there exists an isometry R of $\mathbb{R}^2$, one of at most $4\deg(\varphi)$ possibilities, such that $(b',c') = R(b,c)$ and $(a',\varphi(a')) = R(a,\varphi(a))$. In other words, (a',b',c') is determined by (a,b,c), up to at most $4\deg(\varphi)$ possibilities. Thus (11) follows.

Finally, let

$$P = \{(a,a') \in A \times A \mid a \sim a'\},$$

and apply Theorem 3 to bound to $|I(P, \Gamma \setminus \Gamma_1)|$. Note that

$$|P| = \Theta\left(\binom{n/t}{2}t\right) = \Theta(n^2/t) = \Theta(sn^{1/2}|D|^{1/2})$$

and

$$\begin{aligned}|\Gamma \setminus (\Gamma_0 + \Gamma_1)| \leq \deg(f)|\hat{\Gamma}| &= \Theta\left(\binom{n/t}{2}\binom{n/t}{2}t^2\right)\\ &= \Theta(n^4/t^2)\\ &= \Theta(s^2 n|D|).\end{aligned}$$

We get

$$\begin{aligned}|Q| &\leq 4|I(P,\Gamma \setminus (\Gamma_0 \cup \Gamma_1))| + 4\deg(\varphi)n^3 + O(|\hat{\Gamma}|),\\ &= O_\varepsilon\left((sn^{1/2}|D|^{1/2})^{9/4+2\varepsilon}\right) + 4\deg(\varphi)n^3.\end{aligned}$$

This proves Proposition 7. ◀

Finally, combining the inequalities from Proposition 6 and Proposition 7, we conclude that

$$sn^3 \leq |Q| = O_\varepsilon\left((n|D|)^{9/8+\varepsilon}\right) + 4\deg(\varphi)n^3.$$

Choosing $s > 8\deg(\varphi)$ and rearranging, we get

$$|D| = \Omega_\varepsilon(n^{5/3-\varepsilon'}).$$

This completes the proof of Theorem 2. □

References

1. G. Elekes. On linear combinatorics, I. *Combinatorica*, 17(4):447–458, 1997.
2. G. Elekes and L. Rónyai. A combinatorial problem on polynomials and rational functions. *J. Combin. Theory Ser. A*, 89(1):1–20, 2000. doi:10.1006/JCTA.1999.2976.
3. J. Pach and F. de Zeeuw. Distinct Distances on Algebraic Curves in the Plane. *Combinat., Probab. Comput.*, 26(1):99–117, 2017. doi:10.1017/S0963548316000225.
4. O. E. Raz and Z. Shem-Tov. Expanding polynomials: A genralization of the Eleke–Rónyai theorem to d variables. *Combinatorica*, 40(5):721–748, 2020. doi:10.1007/S00493-020-4041-0.

5 O. E. Raz and J. Solymosi. Dense graphs have rigid parts. *Discrete Cumput. Geom.*, 69:1079–1094, 2023. doi:10.1007/S00454-022-00477-7.

6 M. Sharir and J. Zahl. Cutting algebraic curves into pseudo-segments and applications. *J. Combin. Theory, Ser. A*, 150:1–35, 2017. doi:10.1016/J.JCTA.2017.02.006.

7 O. E. Raz M. Sharir and F. de Zeeuw. The Elekes–Szabó theorem in four dimensions. *Israel J. Math.*, 227:663–690, 2018.

8 O. E. Raz M. Sharir and J. Solymosi. Polynomials vanishing on grids: The Elekes-Rònyai problem revisited. *Amer. J. Math.*, 138(4):1029–1065, 2016.

9 J. Solymosi and J. Zahl. Improved Elekes–Szabó type estimates using proximity. *J. Comb. Theory, Ser. A*, 201:105813, 2024. doi:10.1016/J.JCTA.2023.105813.

10 E. Szemerédi and W. T Trotter. A Combinatorial Distinction Between the Euclidean and Projective Planes. *European Journal of Combinatorics*, 4:385–394, 1983. doi:10.1016/S0195-6698(83)80036-5.

Robustness of Persistent Topological Features and Minimum Homological Cuts

Pepijn Roos Hoefgeest
KTH Royal Institute of Technology Stockholm, Sweden

Lucas Slot
University of Amsterdam, The Netherlands

Abstract

Persistent homology is a popular method for computing topological features of (metric) data. Standard approaches based on the Čech or Rips filtration are stable under small perturbations of the data, but highly sensitive to outliers. This lack of robustness has been frequently addressed in the literature. In this paper, we take a novel perspective by asking the following question: When can we guarantee that an observed persistent feature (a bar) is inherent to the underlying data in the presence of a limited number of unknown, arbitrary outliers. We formalize this question by introducing the notion of *adversarial robustness*, and study the problem of deciding whether a given bar in the barcode of a filtered simplicial complex is adversarially robust. We show that this problem is essentially equivalent to a homological variant of the minimum cut problem in simplicial complexes, which we believe to be of independent interest. As our main technical contribution, we provide the first computational complexity results for this problem, consisting of an efficient algorithm in 0-dimensional homology, NP-hardness for the general problem, and an efficient algorithm for codimension-1 in n-dimensional complexes embedded in $\mathbb{R}^n$. We also analyze its natural linear programming relaxation, whose dual defines a homological analog of the max-flow problem in graphs. We show that a max-flow/min-cut theorem does not hold in our setting, implying that the LP relaxation is not tight in general. Finally, in the special case of the Rips filtration, we provide a global heuristic based on the Hausdorff distance that guarantees adversarial robustness of sufficiently long bars. This connects adversarial robustness to standard stability theorems in persistent homology.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases Topological Data Analysis, Persistent Homology, Min-cut Max-flow, Robustness, Vietoris-Rips Filtration

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.87

Related Version *Full Version*: `https://arxiv.org/abs/2602.23154` [28]

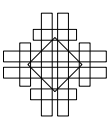
Funding *Pepijn Roos Hoefgeest*: This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.
Lucas Slot: This work was completed while the author was at ETH Zurich, and supported by the Swiss National Science Foundation (SNSF), grant no. 10004947.

1 Introduction

Persistent homology is a central method in topological data analysis. It is used to extract topological features from (metric) data across a range of spatial scales. At a high level, it works as follows. First, we represent our data by a nested sequence of simplicial complexes, called a filtration. An important example is the (Vietoris-)Rips filtration, which arises from pairwise distances between points in a metric space. The evolution of the homology groups of the complexes in this filtration (i.e., its *persistent* homology) can be captured by a *barcode*, which consists of a (multi)set of intervals (or *bars*) whose endpoints represent the



© Pepijn Roos Hoefgeest and Lucas Slot;
licensed under Creative Commons License CC-BY 4.0



appearance (*birth*) and disappearance (*death*) of a homology class at a particular step in the filtration. Barcodes serve as a topological signature of the underlying data set. We refer to [19, 21, 26, 32] for surveys on persistent homology and its many applications.

Stability and robustness. An important property of barcodes is their *stability* under small perturbations: For example, the *Bottleneck distance* between the barcodes of Rips filtrations may be bounded in terms of the *(Gromov-)Hausdorff distance* between their underlying metric spaces [16]. On the other hand, barcodes are (in)famously sensitive to *outliers*. Indeed, even a single outlier may cause arbitrarily large changes in the barcode, which makes persistent homology unreliable in the presence of noise. This issue has been frequently addressed in the literature; we highlight three approaches. First, one may consider alternative metric interpretations of the data which take density into account, and whose resulting barcodes are (hopefully) more robust. For instance, a filtration based on the *distance-to-measure* function [14] achieves stability with respect to the *Wasserstein distance*, which tolerates small amounts of outliers [8]. See [24, 27] for similar approaches using kernel functions. A downside is that these filtrations are difficult to interpret geometrically, often depend on some choice of secondary parameters, and are computationally expensive, making them less suited for topological inference than the Rips filtration. A second approach is to (cleverly) subsample the data to determine so-called *landmarks*, and then construct a filtration based on these landmarks. An example is the *(lazy) witness filtration* [29]. Empirically, it appears that the landmarks may be chosen in a way that reduces the sensitivity to outliers (in the original data); see [30]. However, this effect is hard to quantify theoretically. Third, one may consider filtrations indexed by multiple parameters, modeling for example both scale and density. The persistent homology of such bifiltrations can be provably robust to certain types of noise [4]. However, multiparameter persistence modules are significantly more difficult to represent than their one-parameter counterparts. In particular, they generally do not have a barcode, which is a serious theoretical and practical drawback; see [7].

A new notion of robustness. In this paper, we take a different perspective. Rather than modify existing filtrations to increase their tolerance to noise, we ask the following question.

▶ **Question 1.** *When can we guarantee that an observed persistent feature (a bar) arising from a filtration is inherent to the underlying data in the presence of (adversarial) noise?*

To address this question, we introduce a new notion of outlier-robustness of persistent topological features (bars) arising from a filtration of a simplicial complex, which we call *adversarial robustness*. Before giving a formal definition, it is illustrative to first consider the Rips filtration on a (finite) set of metric data. Intuitively speaking, we say that a persistent feature of the Rips filtration is k-adversarially robust if it "continues to exist" after removing any k points from the data set. In this way, k-adversarial robustness certifies that a feature is *inherent* to an underlying data set even after adding (at most k) unknown, arbitrary outliers. Alternatively, the largest k for which a feature is k-adversarially robust can be thought of as a *measure of robustness* of that feature (possibly after dividing by the total number of data points). With respect to the earlier work outlined above, a key advantage of our approach is that we make no structural assumptions on the outliers.

In what follows, we give a formal definition of adversarial robustness. The key ingredient is the induced matching between barcodes [2], which allows us to relate the persistent features of a filtered simplicial complex to those of its filtered subcomplexes. Then, we study the computational problem of deciding whether a given bar is adversarially robust. To this end,

we first show that adversarial robustness of a bar can be determined by solving a homological variant of the minimum cut problem in a single (unfiltered) simplicial complex. Thus, it suffices to study the complexity of that problem, which we believe to be of independent interest. We give efficient algorithms for zero-dimensional homology and complexes embedded in $\mathbb{R}^n$. On the other hand, we show that the general problem is NP-hard. We also analyze its natural linear programming relaxation, whose dual defines a homological analog of the max-flow problem in graphs. We conclude by giving an efficiently computable heuristic for adversarial robustness in Rips filtrations, which is connected to Hausdorff stability.

1.1 Adversarially robust persistent features

Let K be a (finite) simplicial complex, and let $\mathcal{K} = (K_i)_{0\leq i\leq m}$ be a filtration of K, i.e., a sequence of simplicial complexes $\emptyset = K_0 \subseteq K_1 \subseteq \ldots \subseteq K_m = K$. Throughout, we assume that $\mathcal{K}$ is a *simplex-wise* filtration, i.e., for any $i < m$, the complex K_{i+1} is obtained by adding (at most) a single simplex to K_i. Any filtration can be made simplex-wise by breaking ties (arbitrarily, but so that face relations are preserved) whenever multiple simplices are added at once, which is also what is done in practice. For $p \geq 0$, we denote the p-dimensional persistent homology of $\mathcal{K}$ by $\mathrm{PH}_p(\mathcal{K})$, and we write $\mathcal{B}(\mathrm{PH}_p(\mathcal{K}))$ for its barcode. For $A \subseteq K$, we write $K - A$ for the largest subcomplex of K contained in $K \setminus A$. That is, $K - A$ is obtained from K by removing all simplices that have a face in A. Similarly, we write $\mathcal{K} - A$ for the filtration of $K - A$ given by $(K_i - A)_{0\leq i\leq m}$. There is a natural way to relate the barcodes associated with $\mathcal{K}$ and $\mathcal{K} - A$. Namely, the inclusion $K - A \hookrightarrow K$ induces a map $\mathrm{PH}_p(\mathcal{K} - A) \to \mathrm{PH}_p(\mathcal{K})$. In turn, this map induces a (partial) matching between the respective barcodes, which we denote $\mathcal{X}_{\mathcal{K}-A\hookrightarrow\mathcal{K}}$. This is a special case of the so-called *induced matching*, which plays a crucial role in a proof of the algebraic stability theorem [2]. Generally, the induced matching is *not* functorial, but in the context of simplex-wise filtrations it allows us to unambiguously relate bars in the barcodes of $\mathrm{PH}_p(\mathcal{K})$ and $\mathrm{PH}_p(\mathcal{K} - A)$; see Section 3 for more details. This allows us to give the central definition of this paper. For $s \in \mathbb{N}$, let $K^{(s)} \subseteq K$ denote the set of s-simplices in K.

▶ **Definition 2** (Adversarial robustness). *Let $\mathcal{K}$ be a simplex-wise filtration of a simplicial complex K. Let $p \geq 0$, and let $s \leq p$. A bar $B \in \mathcal{B}(\mathrm{PH}_p(\mathcal{K}))$ is* k-adversarially robust (in degree s) *if, for each subset $A \subseteq K^{(s)}$ of size at most k, we have $B \in \mathrm{Im}(\mathcal{X}_{\mathcal{K}-A\hookrightarrow\mathcal{K}})$.*

Definition 2 guarantees that a bar is "present" in any (filtered) subcomplex of K missing at most k s-simplices. If $\mathcal{K}$ is a Rips filtration, the 0-simplices (vertices) of K correspond to metric data points. Thus, k-adversarial robustness (in degree 0) of a bar means that it is present in the Rips filtration of any subset of the original data obtained by removing at most k points, matching our earlier intuitive description.

1.2 (Minimum) homological cuts

Adversarial robustness of bars is closely related to a novel homological variant of the min-cut problem in simplicial complexes, which we introduce in this work. For $C \subseteq K^{(s)}$, we write $\iota^*_{K-C\hookrightarrow K} : H_p(K - C) \to H_p(K)$ for the map induced by the inclusion $K - C \hookrightarrow K$.

▶ **Definition 3** (homological cuts). *Let K be a simplicial complex. Let $p \geq 0$, and $s \leq p$. We say that $C \subseteq K^{(s)}$ is a* homological s-cut *for $\gamma \in H_p(K)$ if $\gamma \notin \mathrm{Im}(\iota^*_{K-C\hookrightarrow K})$. We refer to the special cases $s = 0, s = 1$ as homological* vertex *and* edge *cuts, respectively; see Figure 1.*

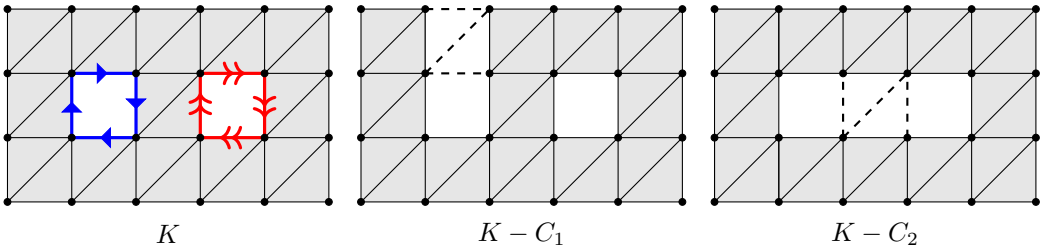


Figure 1 A simplicial complex K with cycles $c_{\text{left}}, c_{\text{right}} \in C_1(K;\mathbb{R})$ drawn in blue single arrows and red double arrows, respectively (all coefficients equal to 1). The classes $[c_{\text{left}}]$ and $[c_{\text{right}}]$ generate $H_1(K;\mathbb{R}) \cong \mathbb{R}^2$. On the right: two subcomplexes obtained by removing subsets $C_1, C_2 \subseteq K^{(1)}$ (dashed) from K, respectively. Note that C_1 is an edge cut for $[c_{\text{left}}]$ and $[c_{\text{left}} + c_{\text{right}}]$, but not for $[c_{\text{right}}]$. On the other hand, C_2 is a 1-cut for both $[c_{\text{left}}]$ and $[c_{\text{right}}]$, but not for $[c_{\text{left}} + c_{\text{right}}]$.

To connect homological cuts to adversarial robustness, we use the fact that, for a simplex-wise filtration $\mathcal{K}$, each bar $B \in \mathcal{B}(\mathrm{PH}_p(\mathcal{K}))$ corresponds to a pair (σ_B, τ_B) of simplices, whose insertions at steps $i = b$ and $i = d$, respectively, represent the birth and death of any cycle representing B. These are called *persistence pairs*. We call the complex $K_B := K_{d-1}$ the *predeath complex* of B. Now, the following proposition shows that adversarial robustness of B is characterized by homological cuts of $[\tau_B]$ in K_B. We give its proof in Section 3.2.

▶ **Proposition 4.** *Let $p \geq 0$, and let $B \in \mathcal{B}(\mathrm{PH}_p(\mathcal{K}))$ be a bar in the barcode of a simplex-wise filtration $\mathcal{K}$ of a simplicial complex K. Let K_B, τ_B be its pre-death complex and death simplex. Then, B is k-adversarially robust (in degree s) if, and only if,*

$$[\partial\tau_B] \in \mathrm{Im}(\iota^*_{K_B - A \hookrightarrow K_B}) \text{ for all } A \subseteq K^{(s)} \text{ with } |A| \leq k.$$

Thus, B is k-adversarially robust iff all homological s-cuts of $[\tau_B]$ in K_B have size at least k.

The above proposition motivates our study of the following problem.

▶ **Problem 5.** *The* minimum homological s-cut problem *asks to compute*

$$\mathrm{mhc}(K, \gamma, s) := \min_{C \subseteq K^{(s)}} \{|C| : C \text{ is a homological } s\text{-cut for } \gamma\}. \tag{MHC}$$

We call a set C attaining the minimum above a minimum homological s-cut *for γ (in K).*

Apart from its connection to adversarial robustness, we believe that the minimum homological cut problem is of intrinsic interest. It is related to, *but distinct from*, two types of well-studied problems in computational topology. On the one hand, there are problems related to finding a smallest (or otherwise "optimal") representing cycle of a homology class in a simplicial complex [3, 5, 12]; this is often referred to as *homology localization*. By contrast, the problem we consider can be thought of as *co*homology localization. The computational complexity of homology localization varies based on the choice of coefficient field, the dimension of the homology group, and additional assumptions on the underlying complex. For example, while the general problem is hard [15], a linear programming relaxation yields an efficient algorithm for finding minimum weight homologous cycles over integer coefficients for complexes whose boundary matrix is totally unimodular [17]. On the other hand, there are ordinary min-cut (and max-flow) problems in graphs with topological structure, e.g., graphs that can be embedded in a surface of low genus. There, (co)homological properties of cuts (or flows) can be used to achieve algorithmic speedups with respect to the general case [10, 11, 22].

Linear programming relaxations. As we explain in the full version of this paper [28], homological edge cuts of a class $\gamma \in H_1(K;\mathbb{R})$ are naturally related to (ordinary) cuts in graphs. Namely, we show that a minimum homological edge cut can be found by optimizing the number $\|\varphi\|_0$ of non-zero coefficients of a vector φ (indexed by the edges of K) under a set of linear constraints involving the boundary matrix of K. The relaxation of this problem obtained by optimizing the 1-norm $\|\varphi\|_1$ instead is a linear program. Its dual may be interpreted as a homological analog of the max-flow problem in graphs. Contrary to the graph setting, we show that there is no max-flow min-cut theorem in our case: homological max-flows are not necessarily integral, meaning the LP relaxation is not tight.

1.3 Main contributions

Complexity of the minimum homological cut problem. We prove positive and negative results on the computational complexity of the minimum homological cut problem (MHC) introduced above, in terms of the number of simplices $|K|$ in the complex K. By Proposition 4, these results have immediate implications on the complexity of determining adversarial robustness of bars. Our first contribution is an efficient algorithm for the case where $\gamma \in H_0(K)$ is a 0-dimensional homology class, i.e., $p = 0$.

▶ **Theorem 6.** *Let K be a simplicial complex , and let $\gamma \in H_0(K)$. We can compute a minimum homological vertex cut for γ in time $O(|K^{(0)}| + |K^{(1)}|)$.*

Theorem 6 follows from the fact that a minimum homological cut of a class in H_0 is always equal to the vertex set of a connected component of K, as we show in Section 4.1.

Next, we show that finding a minimum homological cut is NP-hard, already when $p = 1$.

▶ **Theorem 7.** *For $p = 1$ and $s = 1$, the minimum homological s-cut problem is NP-hard. This is true in particular when homology is taken with coefficients in $\mathbb{R}$ or $\mathbb{Z}/2\mathbb{Z}$.*

We give the proof of Theorem 7 in Section 4.2. It relies on a reduction from Exact Cover by 3-Sets (X3C). In [23, pp. 246], X3C is used to show hardness of the problem of finding minimum weight solutions to linear equations. It was also used in [25] to show hardness of finding the *sparsest* approximate solution to a set of linear equations. As mentioned, we show in the full version of this paper [28] that (MHC) can be solved by finding the sparsest solution to a particular set of linear equations. In light of these observations, X3C is a natural candidate for showing hardness of (MHC). In our proof, we construct for any X3C-instance an equivalent instance of (MHC). Our construction relies on "gluing together" punctured discs, each representing a 3-set of the X3C-instance, to obtain a topological space which is then triangulated carefully to ensure minimum homological cuts correspond to exact covers. This approach resembles earlier work on NP-hardness in homology localization [1, 15, 20]. A distinction is that these mostly rely on reductions from SAT-problems; to the best of our knowledge, a reduction from X3C was not considered before in this context. We remark that our construction can be extended to cover the case $s = 0$. Via Proposition 4, our result also implies that testing k-adversarial robustness of bars (in degree 0 and 1) is hard in general.

Finally, we give an efficient algorithm for the following special case, which covers, e.g., subcomplexes of the Delauney triangulation of a finite metric data set in $\mathbb{R}^n$.

▶ **Theorem 8.** *Suppose that K is an n-dimensional simplicial complex embedded in $\mathbb{R}^n$, and let $\gamma \in H_{n-1}(K)$. We can compute a minimum homological $(n-1)$-cut for γ in polynomial time in the number of simplices of K.*

Our proof of Theorem 8 in Section 4.3 relies on Alexander duality, which can be used to construct the so-called *extended dual graph* of K. In [18], this graph was used to find optimal generators for certain homology classes: these correspond to minimum cuts in the graph. For us, the situation is precisely the opposite: as we will show, minimum homological cuts in K correspond to shortest paths in the extended dual graph (allowing for efficient computation). The use of the extended dual graph in homology localization dates back to [31]; see also [9].

The Rips filtration. Complementing our hardness result, we give an efficiently computable *heuristic* based on the Hausdorff distance to test adversarial robustness of bars arising from the Rips filtration of a metric data set X. Namely, we show that any bar in the barcode of the Rips filtration of length at least

$$\mathcal{H}_{X,k} := \max_{A \subseteq X,\ |A| \leq k} d_H(X \setminus A, X)$$

is automatically k-adversarially robust (in degree 0)[1]; see Section 5 for a precise statement. Moreover, we show that the parameter $\mathcal{H}_{X,k}$ can be computed in time $O(|X|^2 \log |X|)$. This result links adversarial robustness to classical stability results in persistent homology. It shows that our definition is at least as expressive as a naive definition based only on the length of bars and the Hausdorff stability of the Rips filtration. That is, sufficiently long bars are outlier-robust according to our definition (as one would expect). But importantly, our definition is also capable of recognizing relatively short bars as robust; see Figure 2.

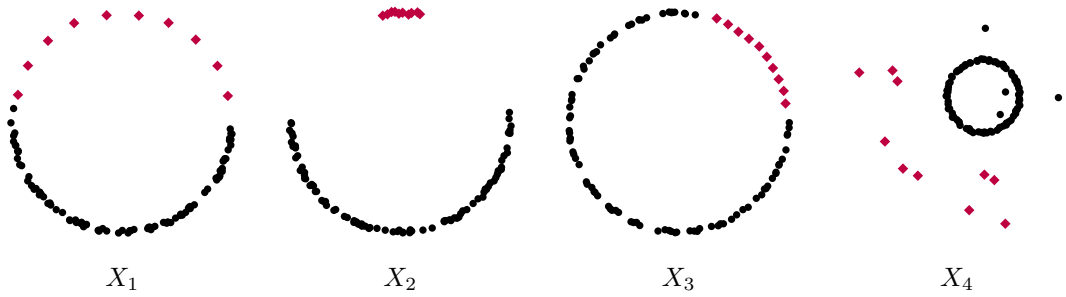


Figure 2 Four data sets in $\mathbb{R}^2$, each of size 100, whose Rips filtrations each induce a 1-dimensional persistent feature. For X_1 and X_2, these features are not k-adversarially robust for $k = 10$, evidenced by the subsets $A_1 \subseteq X_1$ and $A_2 \subseteq X_2$ marked in red. The points in A_2 are quite dense, and so standard subsampling techniques will likely not remove them. On the other hand, the feature is 10-robust in X_3, as its length exceeds $\mathcal{H}_{X_3,10}$, which equals $d_H(X_3 \setminus A_3, X_3)$ for the subset $A_3 \subseteq X_3$ marked in red (cf. Theorem 20). In X_4, the feature induced by the densely sampled circle is 10-robust, even though its bar has length strictly less than $\mathcal{H}_{X_4,10}$ (evidenced by the set A_4 marked in red).

2 Preliminaries

2.1 Simplicial complexes

An (abstract) simplicial complex (on a set Y) is a collection K of non-empty subsets $\sigma \subseteq Y$, called *simplices*, such that if $\tau \in K$ and $\sigma \subseteq \tau$, then $\sigma \in K$. In that case, σ is called a *face* of τ, and τ is called a *coface* of σ. For $A \subseteq K$, we denote by $K - A$ the largest subcomplex

[1] The Rips filtration is not simplex-wise, therefore, as mentioned, we must break ties to obtain a simplex wise filtration to apply Definition 2. This result does not depend on the choice of tiebreaker.

of K which does not contain any simplices in A (which is obtained by removing all simplices from K which have a face in A). For $s \in \mathbb{N}$, we write $K^{(s)}$ for the set of s-simplices of K, being the simplices $\sigma \in K$ with $|\sigma| = s + 1$. We write $V(K) := K^{(0)}$ for the vertices of K, and $E(K) := K^{(1)}$ for its edges. If there is a p such that $K^{(p)} \neq \emptyset$ and $K^{(q)} = \emptyset$ for all $q > p$, then we say that K is p-dimensional, and we call K a p-complex. If K is a p-complex and each simplex of K is contained in a p-simplex, then K is called a *pure* p-complex.

Embedded complexes. A *geometric* p-simplex σ in $\mathbb{R}^n$ is the convex hull of $p+1$ affinely independent vectors $v_0, \ldots, v_p \in \mathbb{R}^n$. The elements of $V(\sigma) := \{v_0, \ldots, v_p\}$ are called the vertices of σ. If τ is the convex hull of a subset of the vertices of σ, then τ is called a *face* of σ. A *geometric* simplicial complex K in $\mathbb{R}^n$ is a collection of geometric simplices in $\mathbb{R}^n$ with the properties that 1) the face of every simplex is in K; 2) for every pair of simplices $\sigma, \tau \in K$ with non-empty intersection, $\sigma \cap \tau$ is in K. We denote the *geometric realization* of K by $\|K\| := \bigcup_{\sigma \in K} \sigma \subseteq \mathbb{R}^n$, i.e. the subset of $\mathbb{R}^n$ consisting of the union of simplices that make up K. If K is a geometric simplicial complex, then $A(K) := \{V(\sigma) \,|\, \sigma \in K\}$ is an abstract simplicial complex. We say that an abstract simplicial complex L is *embedded in* $\mathbb{R}^n$ if $L \cong A(K)$ for some geometric simplicial complex K in $\mathbb{R}^n$, and we define the geometric realization of the embedded simplicial complex L to be $\|L\| := \|K\|$.

Simplicial homology. For a field $\mathbb{F}$, and $p \in \mathbb{N}$, we denote by $C_p(K; \mathbb{F})$ the simplicial p-chains on K. We write $\partial_p : C_p(K; \mathbb{F}) \to C_{p-1}(K; \mathbb{F})$ for the boundary operator. We have subspaces of cycles $Z_p(K; \mathbb{F}) = \mathrm{Ker}(\partial_p)$ and boundaries $B_p(K; \mathbb{F}) = \mathrm{Im}(\partial_{p+1})$, and the simplicial homology of K is defined as $H_p(K; \mathbb{F}) = Z_p(K; \mathbb{F})/B_p(K; \mathbb{F})$. Throughout, if we omit $\mathbb{F}$ from the notation, we mean homology over any field. For two complexes $K \subseteq K'$, we denote the inclusion $K \hookrightarrow K'$ by $\iota_{K \hookrightarrow K'}$. Then, we write $\iota^*_{K \hookrightarrow K'} : H_p(K) \to H_p(K')$, $[c] \mapsto [\iota_{K \hookrightarrow K'}(c)]$ for the induced maps in homology (suppressing p in the notation).

2.2 Persistent homology

A persistence module is a functor $M : (\mathbb{R}, \leq) \to \mathbf{vect}_{\mathbb{F}}$, where $\mathbf{vect}_{\mathbb{F}}$ denotes the category of finitely generated vector spaces. A morphism between persistence modules M and N is a natural transformation of functors $f : M \to N$. There exists a multiset of intervals $\mathcal{B}(M)$, called the barcode of M, that completely captures M up to isomorphism. We write $\mathbb{I}_I$ for the interval module over I, i.e, the persistence module which is equal to $\mathbb{F}$ on I, connected by identity morphisms, and 0 elsewhere. Then there is an isomorphism $M \cong \bigoplus_{I \in \mathcal{B}(M)} \mathbb{I}_I$ [6].

A filtration $\mathcal{K} = (K_i)_{0 \leq i \leq m}$ of simplicial complexes naturally defines a persistence module $\mathrm{PH}_p(\mathcal{K})$, called the persistent homology of $\mathcal{K}$, which is given at $t \in \mathbb{R}$ by $\mathrm{PH}_p(\mathcal{K})_t = \mathrm{H}_p(K_{\lfloor t \rfloor})$, and whose structure maps are maps induced by inclusions. If $\mathcal{K}$ is a simplex-wise filtration, there is at most one birth or death of a bar at each step of the filtration. Hence, any bar $B = [i, j) \in \mathcal{B}(\mathrm{PH}_p(\mathcal{K}))$ then uniquely corresponds to a pair of simplices (σ_B, τ_B), inserted at steps i and j respectively. This is called a *persistence pairing*, and we sometimes refer to the bar as $B = [\sigma_B, \tau_B)$. We denote by $K_B := K_{d-1} = K_d \setminus \{\tau_B\}$ the *predeath complex* of B. For a filtration $\mathcal{K}$ and a non-decreasing function $f : \mathbb{N} \to \mathbb{R}$, we get a persistence module $\mathrm{PH}_p(\mathcal{K}, f)$ given at time $t \in R$ by $\mathrm{H}_p(K_{\sup\{i : f(i) \leq t\}})$. We think of f as a reparametrization of the filtration. Note that if f is the identity, $\mathrm{PH}_p(\mathcal{K}, f) = \mathrm{PH}_p(\mathcal{K})$. For each bar B in the barcode of $\mathrm{PH}_p(\mathcal{K}, f)$ there is a bar $[i, j)$ in the barcode of $\mathrm{PH}_p(\mathcal{K})$ such that $B = [f(i), f(j))$.

3 Induced matchings, adversarial robustness and homological cuts

3.1 Induced matchings

While the barcode of a persistence module M is unique, the isomorphism $M \cong \bigoplus_{I \in \mathcal{B}(M)} \mathbb{I}_I$ is not. For a morphism $f : M \to N$, this makes it difficult to relate the barcodes $\mathcal{B}(M)$ and $\mathcal{B}(N)$ to one another in a way that reflects the algebraic structure of f. In [2], a procedure is described that associates a partial matching $\mathcal{X}_f : \mathcal{B}(M) \nleftrightarrow \mathcal{B}(N)$ with f, in a way that only depends on the barcodes of M, N, and $\mathrm{Im}(f)$, the image persistence module of f. Denote by $\langle \cdot, d \rangle_M \subseteq \mathcal{B}(M)$ the bars in the barcode of the form $[b, d)$ (for simplicity, we assume all intervals in the barcodes of M and N are half-open). If f is injective, it can be shown that $|\langle \cdot, d \rangle_M| \leq |\langle \cdot, d \rangle_N|$. Moreover, if we order the bars in both $\langle \cdot, d \rangle_M$ and $\langle \cdot, d \rangle_N$ by their length, there is an order preserving injection $\mathcal{X}_f^d : \langle \cdot, d \rangle_M \hookrightarrow \langle \cdot, d \rangle_N$, with the property that if $\mathcal{X}_f^d([b, d)) = [b', d)$, then $b' \leq b$. Note that if M and N come from simplex-wise filtrations, the sets $\langle \cdot, d \rangle_M$ and $\langle \cdot, d \rangle_N$ consist of at most one element. The induced matching $\mathcal{X}_f$ is then defined as the union of the matchings $\mathcal{X}_f^d$. If f is surjective, there is a similar way to match bars with the same birth in an order preserving way. For arbitrary $f : M \to N$, there is a factorization $f : M \twoheadrightarrow \mathrm{Im}(f) \hookrightarrow N$ of f into a composition of a surjective and an injective map. The induced matching $\mathcal{X}_f$ is then defined as the composition of the induced matchings of $M \twoheadrightarrow \mathrm{Im}(f)$ and $\mathrm{Im}(f) \hookrightarrow N$. The induced matching is *not* functorial in f.

A persistence module M is called ε-trivial if for each $t \in \mathbb{R}$, the internal morphism $\phi^M_{t \to t+\varepsilon} : M_t \to M_{t+\varepsilon}$ is the zero morphism. The following theorem is the main result of [2].

▶ **Theorem 9** (Induced matching theorem [2]). *Let $f : M \to N$ be a morphism with ε-trivial kernel. Then each bar $[b, d) \in \mathcal{B}(M)$ with $d - b > \varepsilon$ is matched to a bar in $\mathcal{B}(N)$ by $\mathcal{X}_f$. If $\mathrm{coker}(f)$ is ε-trivial, $\mathcal{X}_f$ matches each bar $[b, d) \in \mathcal{B}(N)$ with $d - b > \varepsilon$ to a bar in $\mathcal{B}(M)$.*

We will need this result for our proof of Theorem 20 in Section 5.

3.2 Adversarial robustness and minimum homological cuts

In this section, we show that k-adversarial robustness of a bar can be determined by solving a minimum homological vertex cut problem. That is, we prove Proposition 4.

Proof of Proposition 4. Let $B = [b, d)$ be a bar in the barcode of $\mathrm{PH}_p(\mathcal{K})$, thus with birth simplex σ_B inserted at step b, and death simplex τ_B inserted at time d. Let $K_B = K_{d-1}$ be its predeath complex. Let $A \subseteq K^{(s)}$, and write $\mathcal{X}_{\mathcal{K}-A \hookrightarrow \mathcal{K}}$ for the matching induced by the inclusion $\mathcal{K} - A \hookrightarrow \mathcal{K}$. We show that $B \in \mathrm{Im}(\mathcal{X}_{\mathcal{K}-A \hookrightarrow \mathcal{K}})$ if and only $[\partial \tau_B] \in \mathrm{Im}(\iota^*_{K_B - A \hookrightarrow K_B})$.

Note that any representative cycle for B is homologous to a scalar multiple of $\partial \tau_B$ in $C_p(K_B)$, as $[\partial \tau_B]$ generates the kernel of $H_p(K_B) \to H_p(K_B \cup \{\tau_B\})$. If $B \in \mathrm{Im}(\mathcal{X}_{\mathcal{K}-A \hookrightarrow \mathcal{K}})$, there is thus a bar $B' = [b', d)$ in the barcode of the image module $\mathrm{Im}(\mathrm{PH}_p(\mathcal{K}-A) \to \mathrm{PH}_p(\mathcal{K}))$. Then, any representative cycle $\xi \in C_p(K_{b'} - A) \subseteq C_p(K_{b'})$ for $[b', d)$, when considered as a homology class $[\xi] \in H_p(K_B)$ through the inclusions $K_{b'} - A \hookrightarrow K_{b'} \hookrightarrow K_B$, is in the kernel of $H_p(K_B) \to H_p(K_B \cup \{\tau_B\})$, and is hence homologous to a scalar multiple of $[\partial \tau_B]$.

Conversely, suppose that $[\partial \tau_B]$ is in the image of $H_p(K_B - A) \to H_p(K_B)$, and let $\xi \in C_p(K_B - A)$ be a cycle such that $[\xi] = [\partial \tau_B] \in H_p(K_B)$. Let $\xi_1, \ldots, \xi_k$ denote representative cycles for summands of $\mathrm{Im}(\mathrm{PH}_p(\mathcal{K} - A) \to \mathrm{PH}_p(\mathcal{K}))$ so that the $[\xi_i]$ are linearly independent in $H_p(K_B)$, and $[\xi] = \lambda_1[\xi_1] + \ldots + \lambda_k[\xi_k]$. Since $[\xi]$ is a non-zero element in the kernel of the map

$$\mathrm{Im}\Big(H_p(K_B - A) \to H_p(K_B)\Big) \to \mathrm{Im}\Big(H_p((K_B \cup \{\tau_B\}) - A) \to H_p(K_B \cup \{\tau_B\})\Big),$$

we see that one of the generators ξ_j must be in this kernel, and thus generate a summand isomorphic to $\mathbb{I}_{[b', d)}$ for some $b \leq b' < d$. This shows that B is in the image of $\mathcal{X}_{\mathcal{K}-A \hookrightarrow \mathcal{K}}$. ◀

4 Complexity of the minimum homological cut problem

4.1 An efficient algorithm for zero-dimensional cuts

Let $\gamma \in H_0(K)$. We show that minimum homological vertex cuts for γ have a very particular structure, allowing us to compute them efficiently. Let $\Gamma_1, \dots, \Gamma_q \subseteq V(K)$ be the vertex sets of the connected components of K, and let $c_1 \in V(\Gamma_1), \dots, c_q \in V(\Gamma_q)$. Then, $H_0(K)$ is generated by the classes $[c_1], \dots, [c_q]$, meaning we can write $\gamma = \sum_{i=1}^{q} \lambda_i [c_i]$, with $\lambda_i \in \mathbb{F}$.

▶ **Proposition 10.** *Let $\gamma = \sum_{i=1}^{q} \lambda_i [c_i] \in H_0(K)$ as in the above. A subset $C \subseteq V(K)$ is a homological vertex cut for γ iff $C \supseteq V(\Gamma_\ell)$ for some $1 \leq \ell \leq q$ with $\lambda_\ell \neq 0$. In particular, a homological mincut for γ is of the form $C = V(\Gamma_\ell)$, where $\ell = \arg\min_i \{|V(\Gamma_i)| : \lambda_i \neq 0\}$.*

Proof. Write $V = V(K)$. For any $A \subseteq V$, and any $v \in V \setminus A$ we have $\iota^*_{K-A \hookrightarrow K}([v]) = [v]$. Furthermore, for any $v, w \in V$, we have $[v] = [w]$ (in $H_0(K)$) if and only if v and w lie in the same connected component of K. Since the classes $[v]$, $v \in V \setminus A$, generate $H_0(K - A)$, we thus have $\gamma \in \mathrm{Im}(\iota^*_{K-A \hookrightarrow K})$ if and only if $V \setminus A$ contains a vertex in each connected component Γ_i of K with $\lambda_i \neq 0$. Thus, $\gamma \notin \mathrm{Im}(\iota^*_{K-A \hookrightarrow K})$ (i.e., A is a homological cut) iff A contains all the vertices of at least one connected component Γ_ℓ of K with $\lambda_\ell \neq 0$. ◀

Proof of Theorem 6. To determine a minimum homological vertex cut of a 0-dimensional class in a simplicial complex K, it suffices by Proposition 10 to find its connected components. We can do so in time linear in $|V(K)| + |E(K)|$ using a depth-first search. ◀

4.2 NP-hardness of the general problem

In this section, we prove Theorem 7. That is, we show that the minimum homological cut problem is hard when $p = s = 1$. Our arguments apply for homology taken over any field, and even over $\mathbb{Z}$. We use a reduction from the following NP-hard problem.

▶ **Problem 11** (Exact cover by 3-sets (X3C)). *Given a set U, and a collection $\mathcal{S} \subseteq \mathcal{P}(U)$ of subsets of U, each of size exactly 3, determine whether $\mathcal{S}$ contains an* exact cover *of U, i.e., whether there is an $\mathcal{S}' \subseteq \mathcal{S}$ so that each element of U occurs in* exactly one *element of $\mathcal{S}'$.*

We proceed as follows: To a given instance $(U, \mathcal{S})$ of the X3C problem, we associate a topological space $X = X(U, \mathcal{S})$, together with a specific homology class $\gamma \in H_1(X)$ (Construction 12). Then, we construct a triangulation $K = K(U, \mathcal{S})$ of this space, which satisfies useful technical properties (Lemma 13). Finally, we show that if there exists a minimum cut for γ in K of a particular size, there exists a solution to the X3C instance, and vice versa.

▶ **Construction 12.** *Let $(U, \mathcal{S})$ be an instance of exact cover by 3-sets. For each $u \in U$, denote by $\mathcal{S}_u = \{S_{u,1}, \dots S_{u,m_u}\} \subseteq \mathcal{S}$ the elements of $\mathcal{S}$ containing u. Let F_u denote the surface obtained from a 2-dimensional disk by removing $|\mathcal{S}_u|$ open disks from its interior. Label the outer boundary of F_u by u, and the interior boundary components by the respective elements of $\mathcal{S}$ they correspond to. Choose an orientation for F_u, inducing an orientation on its boundary components. Choose generators for the homology groups of the boundary components such that $\partial[F_u] = [u] + [S_{u,1}] + \dots + [S_{u,m_u}]$; see Figure 3.*

We obtain $X = X(U, \mathcal{S})$ from $\bigsqcup_{u \in U} F_u$ by gluing together the surfaces in the following way: We identify all outer boundary components with each other along orientation preserving homeomorphisms, so that $[u] = [v]$ in $H_1(X)$ for each $u, v \in U$. We also identify all interior boundary components with each other that have the same label, again in an orientation preserving way. This means that if $S = \{u, v, w\}$, then after gluing $S = S_{u,i_u} = S_{v,i_v} = S_{w,i_w}$, and $[S_{u,i_u}] = [S_{v,i_v}] = [S_{w,i_w}]$ in $H_1(X)$. Finally, we set $\gamma = [u] \in H_1(X)$ for some $u \in U$.

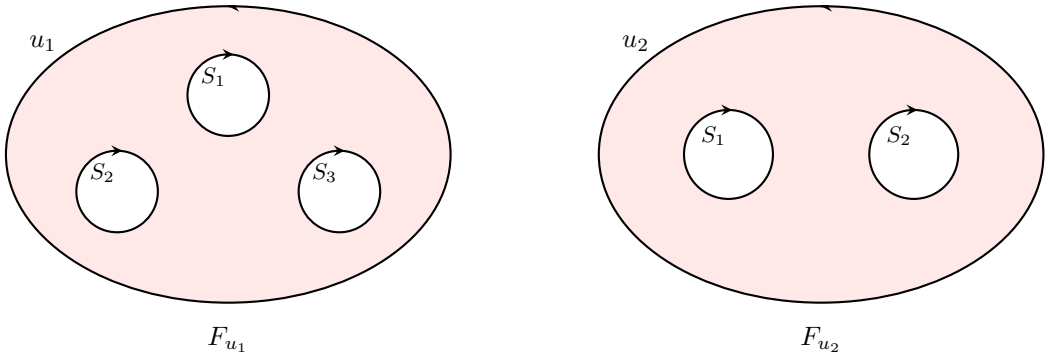


Figure 3 Surfaces F_{u_1} and F_{u_2} involved in the construction of $X(U,\mathcal{S})$ for $(U,\mathcal{S}) = (\{u_1,u_2,u_3,u_4\},\{S_1,S_2,S_3\})$ with $S_1 = \{u_1,u_2,u_3\}, S_2 = \{u_1,u_2,u_4\}, S_3 = \{u_1,u_3,u_4\}$. The surfaces are glued along boundary components with the same label, in an orientation preserving way.

▶ **Lemma 13.** *Let $(U,\mathcal{S})$ be an instance of X3C, and let $k = \max_{u\in U} |\{S \in \mathcal{S} \mid u \in S\}|$. Set $c = c(U,\mathcal{S}) := 5 \cdot \lceil \log_3(k) \rceil + 3$. There exists a triangulation $K = K(U,\mathcal{S})$ for the space $X(U,\mathcal{S})$ of Construction 12, obtained by constructing a triangulation L_u for each surface F_u and gluing them together, that has the following properties:*

1. *K is a pure 2-complex consisting of at most $O(|U| \cdot |S|)$ 2-simplices;*
2. *A minimum homology edge cut for $[u] \in H_1(L_u)$ consists of precisely $c+2$ edges;*
3. *A minimum homology edge cut for $[u]$ contains precisely one edge from u, and precisely one edge from any of the interior boundary components $S_{u,i}$. For any edge in u and any edge in any $S_{u,i}$, there is a minimum cut for $[u]$ containing both those edges.*

Proof. We give an explicit construction in the full version of this paper [28]. ◀

Using Lemma 13, we can sketch our reduction; a complete proof is given in the full version [28].

Proof sketch of Theorem 7. We show that there exists a solution to the X3C-instance $(U,\mathcal{S})$ if and only if there exists a cut for $\gamma \in H_1(K(U,\mathcal{S}))$ consisting of at most $c \cdot |U| + \frac{|U|}{3} + 1$ edges. If $\mathcal{S}^* \subseteq \mathcal{S}$ is a solution to $(U,\mathcal{S})$, for each $u \in U$, we pick a cut C_u for $[u] \in H_1(L_u)$ with the property that it contains one edge in the boundary labelled by u, and one in the boundary labeled by the set $S_u \in S^*$ for which $u \in S_u$. If we consistently choose the same edges in the intersections between surfaces, we can verify that $C = \bigcup_{u\in U} C_u$ is a cut for γ consisting of precisely $c \cdot |U| + \frac{|U|}{3} + 1$ edges. Conversely, suppose C is any cut for γ. Then it can be shown that for each $u \in U$, $C \cap L_u$ is a cut for $[u] \in H_1(L_u)$. By point 3 of Lemma 13, we then obtain a cover $\mathcal{C}_C$ for U. It can be checked that $|C| \geq c \cdot |U| + |\mathcal{C}_C| + 1$. It follows that if $|C| \leq c \cdot |U| + \frac{|U|}{3} + 1$, then $|\mathcal{C}_C| = \frac{|U|}{3}$, and so $\mathcal{C}_C$ is an exact cover. ◀

4.3 An efficient algorithm for embedded complexes

In this section, we prove Theorem 8. To construct an efficient algorithm, we rely on *Alexander duality*. Recall that, for an embedded complex $K \subseteq \mathbb{R}^n$, Alexander duality gives an isomorphism $H_{n-1}(K) \cong \tilde{H}^0(\mathbb{R}^n \setminus \|K\|)$ between $(n-1)$-dimensional homology and 0-dimensional (reduced) singular cohomology. For a subcomplex $L \subseteq K$, a class $\gamma \in H_{n-1}(K)$ lies in the image of $H_{n-1}(L) \to H_{n-1}(K)$ if and only if its Alexander dual lies in the image of the corresponding map $\tilde{H}^0(\mathbb{R}^n \setminus \|L\|) \to \tilde{H}^0(\mathbb{R}^n \setminus \|K\|)$. The latter map can be understood via an extension of the *dual graph* of K.

▶ **Definition 14** (Dual graph). *Let K be a simplicial complex. We denote by $\mathcal{G}_K$ the* dual graph *of K, which has a vertex for each n-simplex of K, and which has an edge between each pair of vertices whose corresponding simplices share an $(n-1)$-dimensional face.*

▶ **Definition 15** (Extended dual graph). *For an embedded simplicial complex K, the extended dual graph $\mathcal{G}_K^*$ of K has one vertex for each n-simplex of K and one vertex for each component of $\mathbb{R}^n \setminus \|K\|$. For each vertex v, we denote by $R_v \subseteq \mathbb{R}^n$ the interior of either the simplex or the connected component of $\mathbb{R}^n \setminus \|K\|$ to which v corresponds. For each $(n-1)$-simplex τ of K, $\mathcal{G}_K^*$ has an edge (v, w) between the vertices for which $\tau \subseteq \overline{R_v} \cap \overline{R_w}$ (possibly allowing loops and multi edges). We denote the set of vertices corresponding to connected components of $\mathbb{R}^n \setminus \|K\|$ by $V_{\mathbb{R}^n \setminus K}$, and the vertex corresponding to the unbounded component by v_∞.*

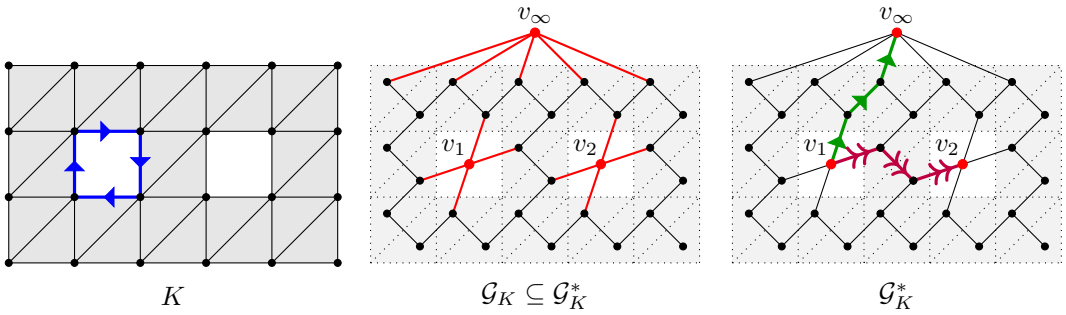


Figure 4 Left: The embedded simplicial complex K of Figure 1 with a class $\gamma \in H_1(K)$ (blue, single arrows). Center: The (extended) dual graph of K. The vertices $V_{\mathbb{R}^n \setminus K} = \{v_1, v_2, v_\infty\}$ and edges added to obtain $\mathcal{G}_K^*$ from $\mathcal{G}_K$ are in red. Only some of the edges incident to v_∞ are drawn. Right: A shortest path $v_1 \to v_\infty$ (green, single arrows) and a shortest path $v_1 \to v_2$ (purple, double arrows) in $\mathcal{G}_K^*$. These correspond to the (minimum) 1-cuts for γ depicted in Figure 1, see Proposition 18. The paths are directed for visual clarity only.

The outline for the rest of this section is as follows. First, we show that one can associate a subgraph $\mathcal{G}_C$ of $\mathcal{G}_K^*$ to any $C \subseteq K^{(n-1)}$ so that $H_{n-1}(K-C) \cong \tilde{H}^0(\mathcal{G}_C)$ (and this association is functorial). In case $C = \emptyset$, $\mathcal{G}_C = \mathcal{G}_K$ and this comes down to Alexander duality, and we see that any $\gamma \in H_{n-1}(K)$ can be written as a sum of homology classes corresponding to vertices in $V_{\mathbb{R}^n \setminus K}$. We then show that whether C is a homology cut for γ depends on the existence of paths in $\mathcal{G}_C$ between specific vertices in $V_{\mathbb{R}^n \setminus K}$. This allows us to reduce (MHC) to a (number of) shortest path problems in $\mathcal{G}_K^*$. See Figure 4 for an illustration.

▶ **Lemma 16.** *Let $K \subseteq \mathbb{R}^n$ be an embedded simplicial complex, and let $C \subseteq K^{(n-1)}$. Denote by $\mathcal{G}_C \subseteq \mathcal{G}_K^*$ the smallest subgraph of $\mathcal{G}_K^*$ containing both $V_{\mathbb{R}^n \setminus K}$ and the dual edges corresponding to the simplices in C. Then there is an isomorphism $H_{n-1}(K-C) \cong \tilde{H}^0(\mathcal{G}_C)$. Moreover, if $C \subseteq C'$, then the following diagram commutes:*

$$\begin{array}{ccc} H_{n-1}(K-C') & \xrightarrow{\cong} & \tilde{H}^0(\mathcal{G}_{C'}) \\ \downarrow & & \downarrow \\ H_{n-1}(K-C) & \xrightarrow{\cong} & \tilde{H}^0(\mathcal{G}_C) \end{array}$$

Proof. See the full version of this paper [28]. ◀

Recall that $H^0(\mathbb{R}^n \setminus \|K\|)$ consists of functions $f : \mathbb{R}^n \setminus \|K\| \to \mathbb{R}$ that are constant on each connected component of $\mathbb{R}^n \setminus \|K\|$. We represent $\tilde{H}^0(\mathbb{R}^n \setminus \|K\|)$ by locally constant functions that are 0 on the unbounded connected component. Similarly, we think of $\tilde{H}^0(\mathcal{G}_C)$

as functions that are constant on the components of $\mathcal{G}_C$, and 0 on the component containing v_∞. Moreover, if $C \subseteq C'$, the map $\tilde{H}^0(\mathcal{G}_{C'}) \to \tilde{H}^0(\mathcal{G}_C)$ is given by a restriction of functions. The following proposition allows us to relate cuts for specific elements of $H_{n-1}(K)$ to paths in the extended dual graph $\mathcal{G}_K^*$ of K.

▶ **Proposition 17.** *Let $v \in V_{\mathbb{R}^n \setminus K} \setminus \{v_\infty\}$. Consider the map $\mathbf{1}_v \in \tilde{H}^0(\mathcal{G}_\emptyset) \cong \tilde{H}^0(\mathbb{R}^n \setminus \|K\|)$. A set $C \subseteq K^{(n-1)}$ is a homological cut for the Alexander dual $\mathrm{AD}(\mathbf{1}_v) \in H_{n-1}(K)$ iff the set of edges in $\mathcal{G}_K^*$ dual to the simplices of C contains a path from v to another vertex in $V_{\mathbb{R}^n \setminus K}$. (Here, $\mathrm{AD}(\mathbf{1}_v)$ is meant to denote the Alexander dual of $\mathbf{1}_{R_v} \in \tilde{H}^0(\mathbb{R}^n \setminus \|K\|)$.)*

Proof. By Lemma 16, $\mathrm{AD}(\mathbf{1}_v)$ is in the image of $H_{n-1}(K-C) \to H_{n-1}(K)$ if and only if $\mathbf{1}_v$ is in the image of $\tilde{H}^0(\mathcal{G}_C) \to \tilde{H}^0(\mathcal{G}_\emptyset)$. This is the case if and only if the function $\mathbf{1}_v$ extends to a function $\varphi \in \tilde{H}^0(\mathcal{G}_C)$, which is possible if and only if v is in a different component from any other vertex of $V_{\mathbb{R}^n \setminus K}$ inside $\mathcal{G}_C$. That means that C is a homology cut for $\mathrm{AD}(\mathbf{1}_v)$ if and only if there is a path in $\mathcal{G}_C$ from v to some other vertex of $V_{\mathbb{R}^n \setminus K}$, proving our claim. ◀

We need the following generalization of the above proposition, that allows us to relate cuts for arbitrary elements of $H_1(K)$ to paths in $\mathcal{G}_K^*$; see Figure 4 for an example.

▶ **Proposition 18.** *Let $\gamma = \sum_{v \in V_{\mathbb{R}^n \setminus K}} \mathrm{AD}(\alpha_v \mathbf{1}_v)$ with $\alpha_{v_\infty} = 0$ be any element of $H_{n-1}(K)$. Then C is a homology cut for γ if and only if the set of edges dual to the simplices of C contains a path between a pair of vertices $v, w \in V_{\mathbb{R}^n \setminus K}$ for which $\alpha_v \neq \alpha_w$.*

Proof. The proof is similar to that of Proposition 17, with the following adjustment: The function $\psi = \sum_{v \in V_{\mathbb{R}^n \setminus K}} \alpha_v \mathbf{1}_v \in \tilde{H}^0(\mathcal{G}_\emptyset)$ extends to a function $\varphi \in \tilde{H}^0(\mathcal{G}_C)$ iff no pair of vertices $v, w \in V_{\mathbb{R}^n \setminus K}$ for which $\psi(v) = \alpha_v \neq \alpha_w = \psi(w)$ is in the same component of $\mathcal{G}_C$. ◀

Proof of Theorem 8. To compute a minimum homological cut for a class $\gamma \in H_{n-1}(K)$ of an n-complex K embedded in $\mathbb{R}^n$, we proceed in three steps:

1. **Constructing the extended dual graph.** First, we construct $\mathcal{G}_K^*$. For this we use the Void Boundary Reconstruction algorithm [18, Section 4], which outputs sets of oriented simplices $(\vec{\xi}_1, \ldots, \vec{\xi}_k)$ that constitute the boundaries of the components of $\mathbb{R}^n \setminus \|K\|$ (cf. Figure 4) in time bounded by $O(|K|^2)$. We assume wlog that $\vec{\xi}_k$ constitutes the boundary of the unbounded component of $\mathbb{R}^n \setminus \|K\|$. The sum of oriented simplices in any of the $\vec{\xi}_i$ with $i < k$ bounding a component $R_i \subseteq \mathbb{R}^n \setminus \|K\|$, which we denote by $[\vec{\xi}_i]$, represents the Alexander dual to the function $f_i \in \tilde{H}^0(\mathbb{R}^n \setminus \|K\|)$ that is constant 1 on R_i, and 0 everywhere else. From the $(\vec{\xi}_1, \ldots, \vec{\xi}_k)$ we can construct $\mathcal{G}_K^*$ in time linear in $k + |K|$. Each $\vec{\xi}_i$ corresponds to a vertex $v_i = v_{R_i} \in V_{\mathbb{R}^n \setminus K} \subseteq V(\mathcal{G}_K^*)$.
2. **Expressing γ in an appropriate basis.** We write γ as $\gamma = \sum_{i=1}^k \alpha_i [\vec{\xi}_i]$ with $\alpha_i \in \mathbb{F}$, $\alpha_k = 0$. This comes down to a change of basis, which can be done in matrix-multiplication time. The representation of γ is precisely the representation as assumed in Proposition 18, as $[\vec{\xi}_i]$ is the Alexander dual of $\mathbf{1}_{v_i}$.
3. **Determining shortest paths.** Using breadth-first search, we can find the minimum distance between any two vertices $v_i, v_j \in V_{\mathbb{R}^n \setminus K}$ inside $\mathcal{G}_K^*$ for which $\alpha_i \neq \alpha_j$ in time bounded by $O(k \cdot |\mathcal{G}_K^*|)$. Let P denote a path in $\mathcal{G}_K^*$ achieving this minimum. The set of $(n-1)$-simplices C_P dual to the edges constituting P form a cut for γ by Proposition 18. Since the edges dual to any cut C for γ must contain a path between vertices of $V_{\mathbb{R}^n \setminus K}$ with differing coefficients, the cut C_P is a minimum cut by minimality of P. ◀

5 Adversarial robustness for the Rips filtration

Let $X = (X, d)$ be a (finite) metric space. For a subset $\sigma \subseteq X$, write $\mathbf{r}(\sigma)$ for its diameter, i.e., the largest pairwise distance between points in σ. The Rips filtration of X is typically defined as the $\mathbb{R}$-indexed family of simplicial complexes $\mathcal{R}(X) := (\{\sigma \subseteq X : \mathbf{r}(\sigma) \leq r\})_{r\in\mathbb{R}}$. If we choose a total order $\preceq$ on the points in X, we can transform $\mathcal{R}(X)$ into a discrete, simplex-wise filtration. Namely, we insert simplices one-by-one according to their diameter, breaking ties first by simplex dimension, and then by the lexicographic order on subsets of X induced by $\preceq$. We denote the resulting *simplex-wise Rips filtration* by $\mathcal{R}^{\preceq}(X)$. The *length* of a bar $B = [\sigma_B, \tau_B)$ in the barcode of $\mathrm{PH}_p(\mathcal{R}^{\preceq}(X))$ is $\ell(B) := \mathbf{r}(\tau_B) - \mathbf{r}(\sigma_B)$.

We consider the following heuristic to determine adversarial robustness of bars in the barcode of $\mathrm{PH}_p(\mathcal{R}^{\preceq}(X))$ based on their length. We denote the Hausdorff distance between $A, B \subseteq X$ by $d_H(A, B) := \max\{\sup_{a\in A} d(a, B), \sup_{b\in B} d(b, A)\}$.

▶ **Definition 19** (Hausdorff heuristic). *Let X be a finite metric space. For $k \in \mathbb{N}$, we define*

$$\mathcal{H}_{X,k} := \max_{A\subseteq X,\, |A|\leq k} d_H(X \setminus A, X). \tag{1}$$

The Hausdorff heuristic has the following two key properties.

▶ **Theorem 20.** *Let $p \geq 0$, and let B be a bar in the barcode of $\mathrm{PH}_p(\mathcal{R}^{\preceq}(X))$. If B has length $\ell(B) \geq \mathcal{H}_{X,k}$, then B is k-adversarially robust in degree 0.*

▶ **Proposition 21.** *The parameter $\mathcal{H}_{X,k}$ can be computed in time $O(|X|^2 \log |X|)$.*

Proof sketch of Theorem 20. Consider the $\mathbb{R}$-indexed Rips filtration $\mathcal{R}(X)$. Let $A \subseteq X$, and set $\delta = d_H(X \setminus A, X)$. For any $r \in \mathbb{R}$, there is a map $\mathrm{H}_p(\mathcal{R}_r(X)) \to \mathrm{H}_p(\mathcal{R}_{r+\delta}(X - A))$ making the following diagram commute:

$$\begin{array}{ccc} H_p(\mathcal{R}_r(X \setminus A)) & \longrightarrow & H_p(\mathcal{R}_{r+\delta}(X \setminus A)) \\ \downarrow & \nearrow & \downarrow \\ H_p(\mathcal{R}_r(X)) & \longrightarrow & H_p(\mathcal{R}_{r+\delta}(X)) \end{array}$$

This can be shown similarly to how stability of Rips persistence is proved, e.g., as in [13]. It follows that the cokernel of $\mathrm{H}_p(\mathcal{R}(X \setminus A)) \to \mathrm{H}_p(\mathcal{R}(X))$ is δ-trivial, which would allow us to apply the induced matching theorem (Theorem 9).

To transport this idea to the simplex-wise Rips filtration $\mathcal{R}^{\preceq}(X)$, we make use of an "ε-smoothening". That is, an $\mathbb{R}$-indexed simplex-wise filtration $\mathcal{R}^{\varepsilon}(X)$, which inserts simplices in the same order as $\mathcal{R}^{\preceq}(X)$, and which is ε-interleaved with $\mathcal{R}(X)$, i.e.,

$$\mathcal{R}^{\varepsilon}_r(X) \subseteq \mathcal{R}_r(X) \subseteq \mathcal{R}^{\varepsilon}_{r+\varepsilon}(X) \quad \forall r \in \mathbb{R}.$$

Such a filtration exists for any $\varepsilon > 0$ small enough. Importantly, the persistence pairings and induced matchings of $\mathcal{R}^{\preceq}(X)$ and $\mathcal{R}^{\varepsilon}(X)$ agree. By the previous, the cokernel of $\mathrm{H}_p(\mathcal{R}^{\varepsilon}(X\setminus A)) \to \mathrm{H}_p(\mathcal{R}^{\varepsilon}(X))$ is $\delta{+}2\varepsilon$-trivial. By Theorem 9, every bar of length greater than $\delta{+}2\varepsilon$ in the barcode of $\mathcal{R}^{\varepsilon}(X)$ is in the image of the matching induced by $\mathcal{R}^{\varepsilon}(X\setminus A) \hookrightarrow \mathcal{R}^{\varepsilon}(X)$. Making ε arbitrarily small, the theorem follows. For details, see the full version [28]. ◀

Proof of Proposition 21. To compute $\mathcal{H}_{X,k}$ efficiently, we use the following structural lemma on the sets A that attain the maximum in (1); for its proof, see the full version [28].

▶ **Lemma 22.** *For $i \in \mathbb{N}$ and $x \in X$, let $\nu_i(x)$ denote the ith nearest neighbor of x. The set $A \subseteq X$ attaining the maximum in (1) is of the form $\{x\} \cup \{\nu_i(x) : i = 1, \ldots, k\}$, with $x \in X$. In particular, we have $\mathcal{H}_{X,k} = \max_{x \in X} d(x, \nu_k(x))$.*

For each $x \in X$, we can compute $\nu_k(x)$ by sorting the points in X by their distance to x, which can be done in $O(|X| \log |X|)$. This immediately proves Proposition 21. ◀

References

1 Ian Agol, Joel Hass, and William Thurston. The computational complexity of knot genus and spanning area. *Transactions of the American Mathematical Society*, 358(9):3821–3850, 2006. doi:10.1090/s0002-9947-05-03919-x.

2 Ulrich Bauer and Michael Lesnick. Induced matchings and the algebraic stability of persistence barcodes. *J. Comput. Geom.*, 6:162–191, 2013.

3 Nello Blaser, Morten Brun, Lars M. Salbu, and Erlend Raa Vågset. The parameterized complexity of finding minimum bounded chains. *Computational Geometry*, 122:102102, 2024. doi:10.1016/j.comgeo.2024.102102.

4 Andrew J. Blumberg and Michael Lesnick. Stability of 2-parameter persistent homology. *Foundations of Computational Mathematics*, 24(2):385–427, 2022. doi:10.1007/s10208-022-09576-6.

5 Glencora Borradaile, William Maxwell, and Amir Nayyeri. Minimum Bounded Chains and Minimum Homologous Chains in Embedded Simplicial Complexes. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2020.21.

6 Magnus Bakke Botnan and William Crawley-Boevey. Decomposition of persistence modules. *Proc. Am. Math. Soc.*, 148(11):4581–4596, 2020.

7 Magnus Bakke Botnan and Michael Lesnick. An introduction to multiparameter persistence. *Preprint, arXiv:2203.14289*, 2023. doi:10.48550/arXiv.2203.14289.

8 Mickaël Buchet, Frédéric Chazal, Steve Y. Oudot, and Donald R. Sheehy. Efficient and robust persistent homology for measures. *Comput. Geom.*, 58:70–96, 2016. doi:10.1016/J.COMGEO.2016.07.001.

9 Chris Buehler, Steven J. Gortler, Michael F. Cohen, and Leonard McMillan. *Minimal Surfaces for Stereo*, pages 885–899. Springer Berlin Heidelberg, 2002. doi:10.1007/3-540-47977-5_58.

10 Erin W. Chambers, Jeff Erickson, Kyle Fox, and Amir Nayyeri. Minimum cuts in surface graphs. *SIAM Journal on Computing*, 52(1):156–195, 2023. doi:10.1137/19M1291820.

11 Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Homology flows, cohomology cuts. *SIAM Journal on Computing*, 41(6):1605–1634, 2012. doi:10.1137/090766863.

12 Erin W. Chambers and Mikael Vejdemo-Johansson. Computing minimum area homologies. *Computer Graphics Forum*, 34(6):13–21, 2015. doi:10.1111/cgf.12514.

13 Frédéric Chazal, David Cohen-Steiner, Leonidas J. Guibas, Facundo Memoli, and Steve Y. Oudot. Gromov-Hausdorff Stable Signatures for Shapes using Persistence. *Computer Graphics Forum*, 2009.

14 Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric inference for probability measures. *Found. Comput. Math.*, 11:733–751, 2011. doi:10.1007/S10208-011-9098-0.

15 Chao Chen and Daniel Freedman. Hardness results for homology localization. *Discrete Comput. Geom.*, 45(3):425–448, 2011. doi:10.1007/S00454-010-9322-8.

16 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37:103–120, 2007. doi:10.1007/S00454-006-1276-5.

17 Tamal K. Dey, Anil N. Hirani, and Bala Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. *SIAM J. Comput.*, 40(4):1026–1044, 2011. doi:10.1137/100800245.

18 Tamal K. Dey, Tao Hou, and Sayan Mandal. Computing minimal persistent cycles: polynomial and hard cases. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, pages 2587–2606, USA, 2020. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611975994.158.

19 Tamal K. Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022.

20 Nathan M. Dunfield and Anil N. Hirani. The least spanning area of a knot and the optimal bounding chain problem. In *Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry*, SoCG '11, pages 135–144, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1998196.1998218.

21 Herbert Edelsbrunner and John Harer. Persistent homology—a survey. *Discrete Comput. Geom.*, 453, 2008.

22 Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proceedings of the 2011 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1166–1176, 2011. doi:10.1137/1.9781611973082.88.

23 Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

24 Pepijn Roos Hoefgeest and Lucas Slot. The Christoffel-Darboux kernel for topological data analysis. In *39th International Symposium on Computational Geometry*, volume 258 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 38, 20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/lipics.socg.2023.38.

25 B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, 1995. doi:10.1137/S0097539792240406.

26 Steve Y. Oudot. *Persistence Theory - From Quiver Representations to Data Analysis*, volume 209 of *Mathematical surveys and monographs*. American Mathematical Society, 2015. URL: http://bookstore.ams.org/surv-209/.

27 Jeff M. Phillips, Bei Wang, and Yan Zheng. Geometric inference on kernel density estimates. In *31st International Symposium on Computational Geometry*, volume 34 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 857–871. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.SOCG.2015.857.

28 Pepijn Roos Hoefgeest and Lucas Slot. Robustness of persistent topological features and minimum homological cuts (full version), 2026. URL: https://arxiv.org/abs/2602.23154.

29 Vin Silva and Gunnar Carlsson. Topological estimation using witness complexes. *Proc. Sympos. Point-Based Graphics*, 2004.

30 Bernadette J. Stolz. Outlier-robust subsampling techniques for persistent homology. *J. Mach. Learn. Res.*, 24:Paper No. [90], 35, 2023.

31 J. Sullivan. *A Crystalline Approximation Theorem for Hypersurfaces*. PhD thesis, Princeton University, 1990.

32 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33:249–274, 2005. doi:10.1007/S00454-004-1146-Y.

Integer Points in Dilates of Polytopes

Shubhangi Saraf
University of Toronto, Canada

Narmada Varadarajan
University of Toronto, Canada

Abstract

In this paper we study how the number of integer points in a polytope grows as we dilate the polytope. We prove new and essentially tight bounds on this quantity by specifically studying dilates of the Hadamard polytope.

The motivation for studying this question comes from the question of understanding the maximal number of monomials in a factor of a multivariate polynomial of s monomials. A recent result by Bhargava, Saraf and Volkovich [4] showed that if f is an n-variate polynomial, where each variable has degree d, and f has s monomials, then any factor of f has at most $s^{O(d^2 \log n)}$ monomials. The key technical ingredient of their proof was to show that any polytope with s vertices, where each vertex lies in $\{0, .., d\}^n$ can have at most $s^{O(d^2 \log n)}$ integer points. The precise dependence on d of the number of integer points was left open. We show that this bound, and in particular the dependence on d is essentially tight by studying dilates of the Hadamard polytope and proving new lower bounds on its number of integer points.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Computational geometry, Complexity theory, Integer polytopes

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.88

Related Version *Full Version*: https://arxiv.org/pdf/2510.16481

Funding Research partially supported by the McLean Award and an NSERC Discovery Grant.

Acknowledgements We thank the reviewers for their helpful suggestions.

1 Introduction

Integer polytopes play a starring role in several areas of mathematics, and have recently found very interesting applications in theoretical computer science. In particular, the problem of counting integer points in polytopes lies at the intersection of combinatorics and convex geometry, with applications in algebraic geometry [7, 8], number theory [3, 6], optimization [2, 1], cryptography [10], and algebraic complexity theory [4], just to name a few. Perhaps the most celebrated result on this topic is the *Ehrhart polynomial*: given an integer polytope $P \subseteq \mathbb{R}^n$, the number of integer points in its *dilate*, $|dP \cap \mathbb{Z}^n|$, is a polynomial in d [9]. When P is an n-dimensional polytope, the leading term of its Ehrhart polynomial is $\mathrm{vol}(P)d^n$, which is a good approximation to the number of integer points for large values of d. However, the coefficients of the Ehrhart polynomial are notoriously difficult to compute (see, for example, [11] for examples of polytopes with *negative* Ehrhart coefficients), so we have little information about integer points when the dilation factor d is much smaller than the dimension n.

Motivation from algebraic complexity theory

Our interest in estimating $|dP \cap \mathbb{Z}^n|$ when d is smaller than n comes from the problem of *sparse polynomial factorization* in complexity theory. A multivariate polynomial $f \in \mathbb{F}[X_1, \ldots, X_n]$ where each variable X_i appears with degree $\leq d$ is *sparse* if it only has poly(n)

© Shubhangi Saraf and Narmada Varadarajan;
licensed under Creative Commons License CC-BY 4.0

monomials. Polynomial factorization is a fundamental question in computational algebra, and sparse polynomial factorization is a special case that has been studied for several decades now [4, 5, 13]. The first randomized algorithm for sparse polynomial factorization exhibited a dependence on the sparsity of the factors of the polynomial [13], which raised the natural question: do factors of sparse polynomials have to be sparse?

We know that this is not true in general: if g is a factor of some polynomial f, there is no polynomial upper bound for the number of monomials of g in terms of the number of monomials of f. In characteristic 0, we have the counterexample $f = \prod_{i=1}^{n}(X_i^d - Y_i^d)$ and $g = \prod_{i=1}^{n}(X_i^{d-1} + X_i^{d-2}Y_i + \cdots + Y_i^{d-1})$. Here, f has 2^n monomials, but g is a factor of f with d^n monomials. In characteristic p, this gap can be more stark. Consider $f = \sum_{i=1}^{n} X_i^p$ and $g = (\sum_{i=1}^{n} X_i)^{\lfloor p/2 \rfloor}$, so f has n monomials, but g is a factor of f with at least $n^{p/2}$ monomials. In general, we do not have know better than the trivial upper bound of d^n for the number of monomials in a factor of a sparse polynomial. However, note that in the setting where we assume that the individual degree d is an absolute constant and n can be arbitrarily large, these two counterexamples no longer exhibit significant sparsity gaps.

The current best-known bound for sparse factorization in this setting is due to [4], who show that if f has s monomials and g is a factor of f, then g has at most $s^{O(d^2 \log n)}$ monomials. The key technique of their proof is to reduce the problem of counting monomials in f and g into one of counting integer points in their Newton polytopes. In this paper we further study the problem of counting integer points in polytopes, since a better upper bound there would directly imply a better bound for the factoring problem. Unfortunately, our results indicate that there is no better upper bound for polytopes, so any attempt to improve the bound for sparsity will require a new approach.

Connections to studying dilates of polytopes

The key technique to proving the upper bound of $s^{O(d^2 \log n)}$ in [4] is to study integer points in polytopes. In particular if $f = g \times h$, then one can show that the Newton polytope of f is the Minkowski sum of the Newton polytopes of g and h. The high-level idea in the proof is to show the following: If g has too many monomials, then the Newton polytope of g must necessarily have many vertices. This can then be used to show that the Newton polytope of f must also have many vertices, hence f must have many monomials. Their main technical contribution is to show that any polytope with s vertices, where each vertex lies in $[-d, +d]^n$, can have at most $s^{O(d^2 \log n)}$ integer points. The precise dependence on d of the number of integer points was left open.

The main goal of our paper is to understand if this bound for integer points in polytopes is tight. Saptharishi observed that the $\log n$ in the exponent of the bound is necessary (see [4] for a proof of this) by counting integer points in *Hadamard polytopes*, a class of simplices with vertices in $\{-1, 1\}^n$. Specifically, he shows that the number of integer points in the Hadamard polytope is at least $n^{\Omega(\log n)}$. In this paper, we show that the dependence of d^2 in the exponent is also essentially tight by studying dilates of the Hadamard polytope.

An astute reader may recognize the Hadamard polytope as the extremal solution to Hadamard's maximal determinant problem: the parallelepiped spanned by its vertices maximizes volume among all parallelepipeds with vertices in $\{-1, 1\}^n$. We expect that the Hadamard polytope should also maximize the number of integer points – a discrete analog of volume – among all simplices with vertices in $\{-1, 1\}^n$. This is the heuristic that motivates us to study the Hadamard polytope as the "worst-case simplex" for our problem.

In the case when $s = \text{poly}(n)$ (as with factors of sparse polynomials), we can reduce our problem about counting integer points in polytopes in $[-d, d]^n$ to counting integer points in dilates of polytopes as follows. First, by Carathéodory's theorem, each vertex of P is a

convex combination of at most $n+1$ points of $\{-d,d\}^n$. Replacing the vertices of P with these points, we obtain a polytope P_d containing P that still has poly(n) vertices, but they now lie in $\{-d,d\}^n$. So, $P_0 = \frac{1}{d}P_d$ is an integer polytope with vertices in $\{-1,1\}^n$ and $|dP_0 \cap \mathbb{Z}^n| \geq |P \cap \mathbb{Z}^n|$.

Our contribution in this paper is to provide lower bounds for the number of integer points in the Hadamard polytope. Our first main result is an elegant algebraic characterization of the integer points in the Hadamard polytope by constructing an explicit bijection with affine subspaces of $\mathbb{F}_2^{\log n}$. This underpins part of our second main result: a lower bound for the number of integer points in small dilates. To lower bound the points in larger dilates, we use the probabilistic method. These bounds show that both the dependences on d^2 and on $\log n$ in the bound $|dP \cap \mathbb{Z}^n| \leq |V(P)|^{O(d^2 \log n)}$ are necessary.

2 Main results

Let $n = 2^m$ for some $m \in \mathbb{N}$. The $n \times n$ *Hadamard matrix* H has its rows and columns indexed by elements of the vector space $\mathbb{F}_2^m$:

$$H(a,b) = (-1)^{\langle a,b\rangle},$$

where $\langle a,b\rangle$ is the standard dot product on $\mathbb{F}_2^m$.

▶ **Definition 1.** *The* Hadamard polytope $P_{\text{Had}} \subset \mathbb{R}^n$ *is the convex hull of the column vectors of* H*.*

The columns of H are linearly independent so they form the vertices of P_{Had}; the *Hadamard polytope* is an $(n-1)$–simplex with vertices in $\{-1,1\}^n$. The lower bound $|P_{\text{Had}} \cap \mathbb{Z}^n| \geq n^{\Omega(\log n)}$ was first observed by Ramprasad Saptharishi and appears in the paper [4]. Our contribution is a matching upper bound on $|P_{\text{Had}} \cap \mathbb{Z}^n|$ through a complete characterization of all integer points in the Hadamard polytope using the vector space structure of F_2^m. This characterization will also be useful when we study integer points in dilates of P_{Had}.

First, some notation: let $\{h_a : a \in \mathbb{F}_2^m\}$ be the set of column vectors of the Hadamard matrix H, i.e. the vertices of the Hadamard polytope P_{Had}. We can also identify the coordinates of $\mathbb{R}^n$ with vectors in $\mathbb{F}_2^m$, with the convention that the first coordinate in $\mathbb{R}^n$ corresponds to the zero vector in $\mathbb{F}_2^m$. For every $a \in \mathbb{F}_2^m$ and $v \in \mathbb{R}^n$, let $v(a)$ denote the ath coordinate of v in the standard basis, and

$$\text{supp}(v) = \{a \in \mathbb{F}_2^m : v(a) \neq 0\}$$

We also know that every point $v \in P_{\text{Had}}$ can be expressed as a convex combination $v = \sum_{a \in \mathbb{F}_2^m} t_a h_a$. Since the vectors $\{h_a : a \in \mathbb{F}_2^m\}$ are linearly independent, this expression is unique, and we can define

$$T(v) = \{a \in \mathbb{F}_2^m : t_a \neq 0\}.$$

This can also be thought of as the support of v in the basis $\{h_a : a \in \mathbb{F}_2^m\}$.

Statements of main results

▶ **Theorem 2.** *For every* $v \in P_{\text{Had}} \cap \mathbb{Z}^n$, supp$(v)$ *is a subspace of* $\mathbb{F}_2^m$*. Further,* $T(v) = \text{supp}(v)^\perp + b$ *for some* $b \in \mathbb{F}_2^m$*, and*

$$v = \frac{1}{|T(v)|} \sum_{c \in T(v)} h_c$$

is actually a uniform linear combination of vertices.

Notice that this theorem is a combinatorial correspondence between affine subspaces of $\mathbb{F}_2^m$ and integer points in P_{Had}. That is, every affine subspace $S \subseteq \mathbb{F}_2^m$, let v_S be the uniform convex combination of the vertices of P_{Had} corresponding to the points of S. In other words, v_S is the barycenter of the corresponding face of P_{Had}. Then, the map $S \to v_S$ is a bijection between affine subspaces of $\mathbb{F}_2^m$ and integer points in P_{Had}.

▶ **Corollary 3.** *The number of integer points in P_{Had} is $|P_{\text{Had}} \cap \mathbb{Z}^n| = n^{\Theta(\log n)}$.*

▶ **Theorem 4.** *For any $d < n \log n$ and fixed $0 < \varepsilon < 1/2$, the number of integer points in the dilate dP_{Had} is lower bounded by*

$$|dP_{\text{Had}} \cap \mathbb{Z}^n| \geq \begin{cases} n^{\Omega(d \log n)}, & d \leq \frac{\log n}{4}; \\ n^{\Omega(\log^2 n)}, & \frac{\log n}{4} \leq d \leq \log^{1.5} n \\ n^{2\varepsilon d^2 / \log n}, & \log^{1.5} n \leq d \leq (n \log n)^{1/2-\varepsilon}; \\ \left(d^{2\varepsilon}/4\right)^n, & (n \log n)^{1/2+\varepsilon} \leq d < n \log n. \end{cases}$$

Proofs of main results

Proof of Theorem 2. First, we need to show that $\text{supp}(v)$ is a subspace of $\mathbb{F}_2^m$. We will actually show something stronger: if $v(a) \neq 0$ and $v(b) \neq 0$, then $v(a+b) = v(a)v(b)$. Note that since v is an integer point in $P_{\text{Had}} \subset [-1,1]^n$, its coordinates actually lie in $\{-1,0,1\}^n$. So, the three cases we need to consider for our proof are: $v(a) = v(b) = 1$; $v(a) = v(b) = -1$; and $v(a) = 1$, $v(b) = -1$. We will write out the details for the case when $v(a) = v(b) = 1$, and the other cases will follow a similar computation.

Since v is a convex combination of the vertices of P_{Had}, we can write it as $v = \sum_{c \in \mathbb{F}_2^m} t_c h_c$, for some nonnegative real numbers $(t_c)_{c \in \mathbb{F}_2^m}$ that satisfy $\sum_{c \in \mathbb{F}_2^m} t_c = 1$. From the definition of the Hadamard matrix H, we get the following identities for the coordinates $a, b \in \mathbb{F}_2^m$,

$$v(a) = \sum_{c \in \mathbb{F}_2^m} t_c h_c(a) = \sum_{c \in \mathbb{F}_2^m} (-1)^{\langle c,a \rangle} t_c = \sum_{c \perp a} t_c - \sum_{c \not\perp a} t_c,$$
$$v(b) = \sum_{c \in \mathbb{F}_2^m} t_c h_c(b) = \sum_{c \in \mathbb{F}_2^m} (-1)^{\langle c,b \rangle} t_c = \sum_{c \perp b} t_c - \sum_{c \not\perp b} t_c.$$

Define

$$s_{ab} = \sum_{c \perp a, c \perp b} t_c; \quad s_a = \sum_{c \perp a, c \not\perp b} t_c; \quad s_b = \sum_{c \not\perp a, c \perp b} t_c; \quad s_0 = \sum_{c \not\perp a, c \not\perp b} t_c.$$

This allows us to rewrite our identity for $v(a)$ as

$$v(a) = s_{ab} + s_a - s_b - s_0.$$

A similar identity for $v(b)$ and the identity $\sum_{c \in \mathbb{F}_2^m} t_c = 1$ gives us the system of linear equations

$$s_{ab} + s_a - s_b - s_0 = v(a) = 1$$
$$s_{ab} - s_a + s_b - s_0 = v(b) = 1$$
$$s_{ab} + s_a + s_b + s_0 = 1$$

The general solution to this system has the form

$$\begin{aligned} s_{ab} - s_0 &= 1 \\ s_a + s_b &= 0 \\ s_b + s_0 &= 0. \end{aligned}$$

Since each of s_{ab}, s_a, s_b, and s_0 must be nonnegative, this yields a unique solution $s_{ab} = 1, s_a = s_b = s_0 = 0$. So,

$$\begin{aligned} v(a+b) &= \sum_{c \in \mathbb{F}_2^m} (-1)^{\langle a+b, c\rangle} t_c \\ &= \sum_{c \perp a, c \perp b} t_c - \sum_{c \perp a, c \not\perp b} t_c - \sum_{c \not\perp a, c \perp b} t_c + \sum_{c \not\perp a, c \not\perp b} t_c \\ &= s_{ab} - s_a - s_b + s_0 \\ &= 1 = v(a)v(b). \end{aligned}$$

The argument for the other cases follows similarly: we set up a system of equations in the variables s_{ab}, s_b, s_a, s_0 that has a unique solution subject to nonnegativity. Computing the value of $v(a+b)$ for this unique solution shows that $v(a+b) = v(a)v(b)$. This shows not only that $\mathrm{supp}(v)$ is a basis, but also that the entries of v are determined by their values on a basis of $\mathrm{supp}(v)$.

Now, let $b_1, \ldots, b_k \in \mathbb{F}_2^m$ be a basis for $\mathrm{supp}(v)$, and choose $\varepsilon_i \in \{0,1\}$ so that $v(b_i) = (-1)^{\varepsilon_i}$ for each $i = 1, \ldots, k$. Consider the linear map $\mathbb{F}_2^m \to \mathrm{supp}(v)$ that sends $c \in \mathbb{F}_2^m$ to $\langle b_1, c\rangle b_1 + \cdots + \langle b_k, c\rangle b_k \in \mathrm{supp}(v)$. The kernel of this map is exactly $\mathrm{supp}(v)^\perp$, so its range must be all of $\mathrm{supp}(v)$. In particular, there is some $b \in \mathbb{F}_2^m$ whose image under this map is $\varepsilon_1 b_1 + \cdots + \varepsilon_k b_k$. So, $v(b_i) = (-1)^{\langle b_i, b\rangle}$ for each $i = 1, \ldots, k$. For any $a \in \mathrm{supp}(v)$, we know that $a = \sum_{i \in I} b_i$ for some $I \subseteq [k]$, so

$$v(a) = \prod_{i \in I} v(b_i) = (-1)^{\sum_{i \in I} \langle b_i, b\rangle} = (-1)^{\langle a, b\rangle}.$$

We claim that $T(v) = \mathrm{supp}(v)^\perp + b$ for this vector b.

Define the vector $v' \in \mathbb{F}_2^m$ by

$$v' = \frac{1}{|\mathrm{supp}(v)^\perp|} \sum_{c \in \mathrm{supp}(v)^\perp + b} h_c.$$

We will show that $v' = v$ by showing that $v'(a) = v(a)$ for every coordinate $a \in \mathbb{F}_2^m$.

First, for each $a \in \mathrm{supp}(v)$ and $c \in \mathrm{supp}(v)^\perp + b$, we have that $\langle a, c\rangle = \langle a, b\rangle$. So,

$$\begin{aligned} v'(a) &= \frac{1}{|\mathrm{supp}(v)^\perp|} \sum_{c \in \mathrm{supp}(v)^\perp + b} h_c(a) \\ &= \frac{1}{|\mathrm{supp}(v)^\perp|} \sum_{c \in \mathrm{supp}(v)^\perp + b} (-1)^{\langle a, c\rangle} \\ &= (-1)^{\langle a, b\rangle} \\ &= v(a). \end{aligned}$$

Next, for each $a \notin \mathrm{supp}(v)$, the $\mathbb{F}_2$-linear map from $\mathrm{supp}(v)^\perp \to \{0,1\}$ that sends $c \in \mathrm{supp}(v)^\perp$ to $\langle a, c\rangle$ is not identically zero. So, the size of the preimage of 0 under this map is

equal to the size of the preimage of 1. This property still holds when we consider the translate $\mathrm{supp}(v)^\perp + b$: the dot product $\langle a, c\rangle$ is 0 for exactly half the vectors $c \in \mathrm{supp}(v)^\perp + b$, and is 1 for the other half. Thus, the average of $h_c(a)$ over all $c \in \mathrm{supp}(v)^\perp + b$ is zero, so

$$v'(a) = \frac{1}{|\mathrm{supp}(v)^\perp|} \sum_{c \in \mathrm{supp}(v)^\perp + b} (-1)^{\langle a,c\rangle} = 0 = v(a),$$

since $a \notin \mathrm{supp}(v)$. This shows that v does indeed have the form we claimed, with $T(v) = \mathrm{supp}(v)^\perp + b$ and

$$v = \frac{1}{|T(v)|} \sum_{c \in T(v)} h_c \qquad \blacktriangleleft$$

Proof of Corollary 3. Note again that the lower bound $|P_{\mathrm{Had}} \cap \mathbb{Z}^n| = n^{\Omega(\log n)}$ was already known by Saptharishi's contribution to [4]; we will provide a matching upper bound of $n^{O(\log n)}$. By Theorem 2, integer points of $P_{\mathrm{Had}} \cap \mathbb{Z}^n$ are in one-to-one correspondence with affine subspaces of $\mathbb{F}_2^m$. Any k-dimensional subspace $V \subseteq \mathbb{F}_2^m$ has exactly 2^{m-k} corresponding affine subspaces, and there are exactly $\begin{bmatrix} m \\ k \end{bmatrix}_2$ (the Gaussian binomial coefficient) subspaces of dimension k. So, using the upper bound $\begin{bmatrix} m \\ k \end{bmatrix}_2 \le 2^{4m^2}$ (see, for example, [12] for a proof of this bound),

$$|P_{\mathrm{Had}} \cap \mathbb{Z}^n| = \sum_{k=0}^{m} 2^{m-k} \begin{bmatrix} m \\ k \end{bmatrix}_2 \le (m+1)2^{m+4m^2} = n^{O(\log n)},$$

which completes the proof. $\blacktriangleleft$

Now we are ready to prove our main theorem: lower bounding the growth of integer points in the dilates dP_{Had}.

Proof of Theorem 4.

▶ **Case 1.** $d \le (\log n)/4$.

Let us introduce some more notation for our counting argument. We will show that there are $n^{\Omega(d \log n)}$ distinct sets of d integer points in P_{Had} whose sums are all distinct in dP_{Had}. For $v \in P_{\mathrm{Had}} \cap \mathbb{Z}^n$, define the *exponent* of v as $\exp(v) = \dim(\mathrm{supp}(v)^\perp)$, so $|T(v)| = 2^{\exp(v)}$, and

$$v = \frac{1}{2^{\exp(v)}} \sum_{a \in T(v)} h_a.$$

Let $t_a(v)$ be $2^{-\exp(v)}$ if $a \in T(v)$, and 0 otherwise; that is, $t_a(v)$ is the coefficient of the vertex h_a in the convex combination for v. Look at all sets $\{v_1, \ldots, v_d\}$ such that

- **(i)** $T(v_i)$ is a subspace, i.e. $T(v_i) = \mathrm{supp}(v_i)^\perp$;
- **(ii)** $\frac{\log n}{2} - d < \exp(v_i) < \frac{\log n}{2} + d$;
- **(iii)** $\exp(v_i) \ne \exp(v_j)$ if $i \ne j$.

Since $d \le \frac{\log n}{4}$, for n large enough, the number of subspaces of $\mathbb{F}_2^{\log n}$ with dimension k for some $k \in \left(\frac{\log n}{2} - d, \frac{\log n}{2} + d\right)$ is $\begin{bmatrix} \log n \\ k \end{bmatrix}_2 \ge 2^{k(\log n - k)} = n^{\Omega(\log n)}$. So, the number of choices for the set of subspaces $\{T(v_1), \ldots, T(v_d)\}$ with distinct exponents in $\left(\frac{\log n}{2} - d, \frac{\log n}{2} + d\right)$ is $n^{\Omega(d \log n)}$. We will now show that each set $\{v_1, \ldots, v_d\}$ satisfying these properties produces a distinct point in $dP_{\mathrm{Had}} \cap \mathbb{Z}^n$, proving the lower bound $|dP_{\mathrm{Had}} \cap \mathbb{Z}^n| \ge n^{\Omega(d \log n)}$.

Now, suppose $\{v_1, \ldots, v_d\}$ and $\{u_1, \ldots, u_d\}$ are two distinct sets satisfying the above conditions. And suppose for contradiction that $\sum_i v_i = \sum_i u_i$. Expanding this in terms of the vertices of P_{Had}, we get

$$\sum_{i=1}^{d} \sum_{a \in \mathbb{F}_2^m} t_a(v_i) h_a = \sum_{i=1}^{d} \sum_{a \in \mathbb{F}_2^m} t_a(u_i) h_a$$

Since the vertices of P_{Had} are linearly independent, we must have the following stronger identity for all $a \in \mathbb{F}_2^m$:

$$\sum_{i=1}^{d} t_a(v_i) = \sum_{i=1}^{d} t_a(u_i).$$

For each fixed a, since $t_a(v_i)$ is either zero or $2^{-\exp(v_i)}$ and the exponents are all distinct, we can think of each sum as the unique binary representation of some dyadic rational number.

When $a = 0$, we get $t_0(v_i) = 2^{-\exp(v_i)}$ and $t_0(u_i) = 2^{-\exp(u_i)}$ since $T(v_i)$ and $T(u_i)$ are linear subspaces containing 0, for every $i = 1, \ldots, d$. This gives us the identity

$$\sum_{i=1}^{d} 2^{-\exp(v_i)} = \sum_{i=1}^{d} 2^{-\exp(u_i)}.$$

So, the sets $\{\exp(v_i) : i \leq d\}$ and $\{\exp(u_i) : i \leq d\}$ must be equal since the above sums represent the same binary number. This means that the only way for the sums $\sum_{i=1}^{d} v_i$ and $\sum_{i=1}^{d} u_i$ to be equal at the 0th coordinate is if their exponents are all the same. Now we will find another coordinate where they differ.

Assume without loss of generality that $\exp(v_i) = \exp(u_i)$ for all $i = 1, \ldots, d$, and $\exp(v_1) < \cdots < \exp(v_d)$. There must be some k such that $T(v_k) \neq T(u_k)$. Choose $a \in T(v_k) \setminus T(u_k)$ (which is possible since they are different subspaces of the same dimension), and now look at the identity

$$\sum_{i=1}^{d} t_a(v_i) = \sum_{i=1}^{d} t_a(u_i).$$

Again, each sum is the unique binary representation of some dyadic rational number, but $t_a(v_k) \neq 0$ while $t_a(u_k) = 0$, so they cannot represent the same number, which is the desired contradiction.

▶ **Case 2.** $(\log n)/4 \leq d \leq \log^{1.5} n$.

We simply reuse the bound from case 1 (note that the lower bound is monotone in d and hence we can just use the bound in the setting $d = \frac{\log n}{4}$),

$$|dP_{\text{Had}} \cap \mathbb{Z}^n| \geq \left| \frac{\log n}{4} P_{\text{Had}} \cap \mathbb{Z}^n \right| \geq n^{\Omega(\log^2 n)}.$$

▶ **Case 3.** $d \geq \log^{1.5} n$.

For this final case we will use the probabilistic method. Take the projection $\mathbb{R}^n \to \mathbb{R}^{n-1}$ that deletes the first coordinate of every vector, and let P_0 be the image of P_{Had} under this map. Recall that $P_{\text{Had}} \subseteq \mathbb{R}^n$ is an $(n-1)$-simplex whose vertices all have first coordinate equal to 1, since the first coordinate corresponds to the zero vector in $\mathbb{F}_2^m$. So, this projection is a bijection on P_{Had}. In particular, integer points of dP_0 are in one-to-one correspondence with integer points of dP_{Had}, so $|dP_{\text{Had}} \cap \mathbb{Z}^n| = |dP_0 \cap \mathbb{Z}^{n-1}|$.

The new polytope P_0 is now a full-dimensional simplex in $[-1,1]^{n-1}$. Since the vertices of the original Hadamard polytope $P_{\text{Had}} \subseteq \mathbb{R}^n$ were orthogonal, for each vertex $v \in V(P_0) \subseteq \mathbb{R}^{n-1}$, the face of P_0 that does not contain v is defined by the hyperplane $\{x \in \mathbb{R}^{n-1} : \langle x, v\rangle = -1\}$. In particular, we can write the simplex P_0 as an intersection of half-spaces:

$$P_0 = \bigcap_{v \in V(P_0)} \{x \in \mathbb{R}^{n-1} : \langle x, v\rangle \geq -1\},$$

and

$$dP_0 = \bigcap_{v \in V(P_0)} \{x \in \mathbb{R}^{n-1} : \langle x, v\rangle \geq -d\}.$$

Fix some set of c coordinates $S \subset [n]$ and some $D \in [d]$ (with c and D to be determined later) and look at the discrete hypercube $H_S[D]$ supported on those coordinates,

$$H_S[D] = ([-D, D] \setminus \{0\})^S \times \{0\}^{[n]\setminus S}.$$

We will use the probabilistic method to show that P_0 contains at least $1/2$ the points of $H_S[D]$ for every choice of S. Note that the hypercubes $H_S[D]$ are disjoint by construction, so

$$|dP_0 \cap \mathbb{Z}^n| \geq \sum_{S \subseteq [n], |S|=c} |dP_0 \cap H_S[D]|.$$

Fix a vertex $v \in V(P_0)$, and a hypercube $H_S[D]$. Let X be a uniform random variable taking values in $H_S[D]$. Since v is a ± 1-valued vector, $\mathbb{E}|\langle X, v\rangle| = 0$. Hoeffding's inequality implies that

$$\mathbb{P}(|\langle X, v\rangle| > d) \leq 2e^{-d^2/2cD}.$$

By the union bound,

$$\mathbb{P}(X \notin dP_0) \leq \sum_{v \in V(P_0)} \mathbb{P}(|\langle X, v\rangle| > d) \leq 2ne^{-d^2/2cD}.$$

Assuming that $2cD \leq \frac{d^2}{2\log n}$, the above probability is at most $1/2$, so at least $1/2$ the points of $H_S[D]$ must be contained in dP_0. Now, summing over the disjoint sets $H_S[D]$ for all $\binom{n}{c}$ choices of S,

$$|dP_{\text{Had}} \cap \mathbb{Z}^n| = |dP_0 \cap \mathbb{Z}^{n-1}| \geq \sum_{S \subseteq [n], |S|=c} |H_S[D] \cap dP_0| \geq \frac{1}{2}\binom{n}{c}(2D)^c.$$

We divide our analysis into two subcases.

▶ **Case 3a.** *$d \leq (n \log n)^{\frac{1}{2}-\varepsilon}$, which implies that $\frac{d^2}{\log n} \leq n^{1-2\varepsilon}$*

In this case, set $D = 1$ and $c = \frac{d^2}{4\log n}$, so

$$\begin{aligned}
|dP_{\text{Had}} \cap \mathbb{Z}^n| &\geq \frac{1}{2}\binom{n}{c}(2D)^c && \text{Approximating the binomial coefficient} \\
&\geq \left(\frac{n}{c}\right)^c && \\
&\geq \left(\frac{n}{n^{1-2\varepsilon}}\right)^{d^2/\log n} && \text{Since } c \leq \tfrac{d^2}{\log n} \leq n^{1-2\varepsilon} \\
&\geq n^{2\varepsilon d^2/\log n} &&
\end{aligned}$$

▶ **Case 3b.** $d \geq (n \log n)^{\frac{1}{2}+\varepsilon}$.

For our probabilistic argument to work, we need $2cD \leq \frac{d^2}{2 \log n}$, which rearranges to $D \leq \frac{d^2}{4c \log n}$. Set $D = d^{2\varepsilon}/4$ and $c = n$. Now,

$$\begin{aligned} \frac{d^2}{4c \log n} &= \frac{d^2}{4n \log n} && \text{Subsituting } n \log n \leq d^{2/(1+2\varepsilon)} \\ &\geq \frac{d^2}{4d^{2/(1+2\varepsilon)}} && \\ &= \frac{d^{4\varepsilon/(1+2\varepsilon)}}{4} && \text{Since } \varepsilon \leq 1/2 \\ &\geq \frac{d^{2\varepsilon}}{4} = D && \end{aligned}$$

as desired. Now,

$$|dP_{\text{Had}} \cap \mathbb{Z}^n| \geq \frac{1}{2}\binom{n}{c}(2D)^c \geq \left(d^{2\varepsilon}/4\right)^n. \qquad \blacktriangleleft$$

References

1. Divesh Aggarwal, Antoine Joux, Miklos Santha, and Karol Węgrzycki. Polynomial time algorithms for integer programming and unbounded subset sum in the total regime, 2024. Submitted July 7, 2024; revised July 11, 2024. doi:10.48550/arXiv.2407.05435.
2. Iskander Aliev and Martin Henk. Feasibility of integer knapsacks. *SIAM Journal on Optimization*, 20(6):2978–2993, 2010. doi:10.1137/090778043.
3. Matthias Beck and Sinai Robins. *Computing the Continuous Discretely*. Undergraduate Texts in Mathematics. Springer, New York, NY, 1 edition, 2007. doi:10.1007/978-0-387-46112-0.
4. Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Deterministic factorization of sparse polynomials with bounded individual degree. *J. ACM*, 67(2), May 2020. doi:10.1145/3365667.
5. P. Bisht and I. Volkovich. On solving sparse polynomial factorization related problems. *Computational Complexity*, 34(7), 2025. doi:10.1007/s00037-025-00268-5.
6. Sheng Chen, Nan Li, and Steven V. Sam. Generalized ehrhart polynomials. *Transactions of the American Mathematical Society*, 364(1):551–569, 2012. doi:10.1090/S0002-9947-2011-05494-2.
7. David A. Cox. Recent developments in toric geometry, 1996. arXiv:alg-geom/9606016.
8. Luis Crespo, Álvaro Pelayo, and Francisco Santos. Ewald's conjecture and integer points in algebraic and symplectic toric geometry, 2024. arXiv:2310.10366.
9. Eugène Ehrhart. Sur les polyèdres rationnels homothétiques à n dimensions. *Comptes rendus de l'Académie des Sciences*, 254:616–618, 1962.
10. Yansong Feng, Hengyi Luo, Qiyuan Chen, Abderrahmane Nitaj, and Yanbin Pan. Computing asymptotic bounds for small roots in coppersmith's method via sumset theory. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025*, pages 3–32, Cham, 2025. Springer Nature Switzerland. doi:10.1007/978-3-032-01855-7_1.
11. Takayuki Hibi, Akihiro Higashitani, Akiyoshi Tsuchiya, and Koutarou Yoshida. Ehrhart polynomials with negative coefficients. *Graphs and Combinatorics*, 35(1):363–371, 2019. doi:10.1007/S00373-018-1990-9.
12. Peter M. Neumann and Cheryl E. Praeger. Cyclic matrices over finite fields. *Journal of the London Mathematical Society, Series 2*, 52(2):263–284, 1995. doi:10.1112/jlms/52.2.263.
13. Joachim von zur Gathen and Erich Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985. doi:10.1016/0022-0000(85)90044-3.

Approximating Convex Hulls via Range Queries

Thomas Schibler
University of California, Santa Barbara, CA, USA

Jie Xue
New York University Shanghai, China

Jiumu Zhu
New York University Shanghai, China

Abstract

Recently, motivated by the rapid increase of the data size in various applications, Monemizadeh [APPROX'23] and Driemel, Monemizadeh, Oh, Staals, and Woodruff [SoCG'25] studied geometric problems in the setting where the only access to the input point set is via querying a range-search oracle. Algorithms in this setting are evaluated on two criteria: (i) the number of queries to the oracle and (ii) the error of the output. In this paper, we continue this line of research and investigate one of the most fundamental geometric problems in the oracle setting, i.e., the convex hull problem.

Let P be an unknown set of points in $[0,1]^d$ equipped with a range-emptiness oracle. Via querying the oracle, the algorithm is supposed to output a convex polygon $C \subseteq [0,1]^d$ as an estimation of the convex hull $\mathcal{CH}(P)$ of P. The error of the output is defined as the volume of the symmetric difference $C \oplus \mathcal{CH}(P) = (C \setminus \mathcal{CH}(P)) \cup (\mathcal{CH}(P) \setminus C)$. We prove tight and near-tight tradeoffs between the number of queries and the error of the output for different variants of the problem, depending on the type of the range-emptiness queries and whether the queries are non-adaptive or adaptive.

- **Orthogonal emptiness queries in d-dimensional space:**
  We show that the minimum error a deterministic algorithm can achieve with q queries is $\Theta(q^{-1/d})$ if the queries are non-adaptive, and $\Theta(q^{-1/(d-1)})$ if the queries are adaptive. In particular, in 2D, the bounds are $\Theta(1/\sqrt{q})$ and $\Theta(1/q)$ for non-adaptive and adaptive queries, respectively.
- **Halfplane emptiness queries in 2D:**
  We show that the minimum error a deterministic algorithm can achieve with q queries is $\Theta(1/\sqrt{q})$ if the queries are non-adaptive, and $\widetilde{\Theta}(1/q^2)$ if the queries are adaptive. Here $\widetilde{\Theta}(\cdot)$ hides logarithmic factors.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases convex hull, range searching

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.89

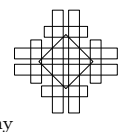
Related Version *Full Version*: https://arxiv.org/abs/2603.20943

1 Introduction

Classic algorithms are designed to compute solutions to a problem instance by processing the *entire* input dataset. These algorithms can suffer two potential drawbacks. The first drawback concerns *efficiency*. As all of the input data has to be received and examined, the time complexity of such algorithms (even the most efficient ones) is at least linear in the input size. However, due to the rapidly growing size of the datasets involved in real-world applications nowadays, linear running time is already not satisfactory in many scenarios. The second drawback regards *data privacy*. For security reasons, sometimes the users would like to keep their data private and therefore cannot directly provide the exact dataset to the algorithm. Instead, they can only provide partial and implicit information of the dataset and ask the algorithm to give useful results based on the information provided. In this situation, the classic algorithms no longer work due to lack of full information of the dataset.



© Thomas Schibler, Jie Xue, and Jiumu Zhu;
licensed under Creative Commons License CC-BY 4.0





Motivated by removing these drawbacks, researchers have studied algorithms in the *oracle model*. In this model, the algorithm does not have direct access to the input dataset. Instead, a provided oracle built on the dataset can answer certain queries about the dataset. The algorithm is required to compute a good solution by performing queries to the oracle. Algorithms in this setting are usually evaluated on two criteria: (i) the number of queries to the oracle and (ii) the error of the output.

For geometric problems, various types of oracles have been considered in the literature [14, 16, 15, 17, 22]. When the input dataset is a set P of points in a Euclidean space $\mathbb{R}^d$, a natural type is the *range-search* oracle. A query to a range-search oracle is a *range* Q in the space $\mathbb{R}^d$ of a specific shape, and the oracle will return certain information about the points in $P \cap Q$. For example, a *range-counting* oracle returns $|P \cap Q|$ [1, 12], a *range-emptiness* oracle returns whether $P \cap Q = \emptyset$ or not [19, 26], a *range-reporting* oracle returns the set $P \cap Q$ itself [6, 23], etc. Range-search oracles have the following advantages. First, range search can usually be implemented very efficiently. As a fundamental topic in Computational Geometry, range search has been extensively studied over decades and many efficient data structures have been proposed for various range queries [4, 5, 9, 12, 13, 21]. Furthermore, most types of range search (except range reporting) do not reveal the exact data points inside the query range Q, and therefore data privacy is well guaranteed. Recently, Monemizadeh [22] and the authors of [17] studied multiple geometric problems with range-search oracles, following earlier work of Czumaj and Sohler [16] and the authors of [15]. Problems considered include facility location [22], clustering [16], Euclidean minimum spanning tree [15, 17], earth mover distance [17], etc. They show that for all these problems, one can obtain nontrivial approximation solutions via a small number of range-search queries.

In this paper, we continue this line of research and investigate one of the most fundamental geometric problems, the *convex hull* problem [7, 8, 10, 11, 24, 27], in the range-search oracle model. In the convex hull problem, the goal is to compute the *convex hull* of a set P of points in $\mathbb{R}^d$, denoted by $\mathcal{CH}(P)$, which by definition is the smallest convex body in $\mathbb{R}^d$ containing P. We study the problem with the simplest type of range-search oracles, i.e., *range-emptiness* oracles. We are interested in finding a convex body C^* as an approximation of $\mathcal{CH}(P)$ via a small number of range-emptiness queries. To this end, we need a measure for the error of the approximation. The most natural measure one can use is the relative error $\|C^* \oplus \mathcal{CH}(P)\| \,/\, \|\mathcal{CH}(P)\|$, where $\|\cdot\|$ denotes the volume and $C^* \oplus \mathcal{CH}(P) = (C^* \backslash \mathcal{CH}(P)) \cup (\mathcal{CH}(P) \backslash C^*)$ is the symmetric difference between C^* and $\mathcal{CH}(P)$. Unfortunately, one can easily see that it is impossible to approximate the convex hull with any bounded relative error no matter how many range-emptiness queries the algorithm performs[1]. Therefore, we shall instead consider the additive error $\|C^* \oplus \mathcal{CH}(P)\|$. Clearly, the additive error depends on the extent measure of P. So we need an extra normalization assumption: we require all points in P to lie in the unit hypercube $[0,1]^d$. Below we formulate the problem to be studied.

Approximate Convex Hull in $\mathbb{R}^d$ via range queries

Input: A (black-box) range-emptiness oracle $\mathcal{O}$ on a set P of points in $[0,1]^d$
Output: A convex body C^* that approximates $\mathcal{CH}(P)$

[1] Even in $\mathbb{R}^1$, there is no way to check whether P only contains one point, in which case $\|\mathcal{CH}(P)\| = 0$, or P contains at least two points, in which case $\|\mathcal{CH}(P)\| > 0$, via range-emptiness queries.

Regarding the above problem, a natural question concerns the tradeoff between the number of queries and the (additive) error of the output: if the algorithm is allowed to perform q queries to the oracle $\mathcal{O}$, what is the minimum error it can achieve (in the worst case)? The answer to this question depends on the following two features of the queries.

- **Shape of the query ranges.** Range queries with different shapes might behave very differently. The most commonly used queries are orthogonal queries [12], where the query ranges are axis-parallel rectangles/boxes. Besides these, well-studied range queries include halfplane/halfspace queries [2, 18], simplex queries [20], semi-algebraic queries [5, 26], etc.
- **Adaptivity of the queries.** In the non-adaptive query model, the algorithm must make all its queries at once, and then make its estimation based on the batched answer. In the adaptive query model, the algorithm is allowed to make queries at any time and the oracle will provide the answer immediately. In particular, the next query can be made after seeing the answers of the previous queries.

As the main contribution of this paper, we prove tight and near-tight tradeoffs between the number of queries and the error of the output for Approximate Convex Hull with orthogonal emptiness queries in $\mathbb{R}^d$ for any fixed d and halfplane emptiness queries in $\mathbb{R}^2$, in both non-adaptive and adaptive query models.

- **Orthogonal queries.** We show the minimum error a deterministic algorithm can achieve for Approximate Convex Hull in $\mathbb{R}^d$ with q orthogonal emptiness queries is $\Theta(q^{-1/d})$ if the queries are non-adaptive and is $\Theta(q^{-1/(d-1)})$ if the queries are adaptive. In particular, in 2D, the bounds are $\Theta(1/\sqrt{q})$ and $\Theta(1/q)$ for non-adaptive and adaptive queries, respectively.
- **Halfplane queries.** We show the minimum error a deterministic algorithm can achieve for Approximate Convex Hull in $\mathbb{R}^2$ with q halfplane emptiness queries is $\Theta(1/\sqrt{q})$ if the queries are non-adaptive and is $\widetilde{\Theta}(1/q^2)$ if the queries are adaptive[2].

Table 1 summarizes the tradeoffs we prove in this paper. All of our algorithms are deterministic and have offline time complexity (outside the oracle) polynomial in q. Our lower bounds similarly hold for deterministic algorithms.

Table 1 Minimum error one can achieve for Approximate Convex Hull with q queries.

Query shape	Space	Query type	Upper bound	Lower bound	Source
Orthogonal	$\mathbb{R}^d$	Non-adaptive	$O(q^{-1/d})$	$\Omega(q^{-1/d})$	Theorems 2 and 8
		Adaptive	$O(q^{-1/(d-1)})$	$\Omega(q^{-1/(d-1)})$	Theorems 12 and 13
Halfplane	$\mathbb{R}^2$	Non-adaptive	$O(1/\sqrt{q})$	$\Omega(1/\sqrt{q})$	Theorems 17 and 22
		Adaptive	$\widetilde{O}(1/q^2)$	$\Omega(1/q^2)$	Theorems 28 and 29

Related work. Both convex hulls and range search have been extensively studied in the literature. See [3, 25] for surveys of these topics. Problems related to convex hulls have also been considered in oracle models prior to this paper. For example, the celebrated work of Chazelle, Liu, and Magen [14] considered the problem of approximating the volume of the convex hull in 2D and 3D via a sampling oracle, which can uniformly sample an input point.

[2] The notation $\widetilde{\Theta}(\cdot)$ hides factors logarithmic in q.

Czumaj and Sohler [16] studied the problem of testing convex position via range queries. Here the goal is to check whether one can remove an ε-fraction of the points from the input point set to make it in convex position. The oracle used in [16] is slightly stronger than range-emptiness oracles – it can report one point in the query range.

2 Preliminaries

Basic notations. For a number $n \in \mathbb{N}$, we write $[n] = \{1, \ldots, n\}$. For a convex body C in $\mathbb{R}^d$, we denote by $\|C\|$ its volume. The notation $|\cdot|$ is used with different meanings depending on the context. For a segment σ, we use $|\sigma|$ to denote the length of σ. For an angle α, $|\alpha| \in [0, 2\pi)$ is the magnitude of α. Furthermore, for a vector $\vec{v}$, $|\vec{v}|$ denotes its magnitude.

Vectors and halfspaces. For a halfspace $H \subset \mathbb{R}^d$, denote by ∂H its bounding hyperplane. (More generally, ∂C denotes the boundary of any convex body C.) The *normal vector* of a halfspace H is a unit vector $\vec{v} \in \mathbb{S}^{d-1}$ perpendicular to ∂H so that for any $q \in \partial H$, $H = \{p \in \mathbb{R}^d : \langle p, \vec{v}\rangle \leq \langle q, \vec{v}\rangle\}$.

3 Approximating convex hulls via orthogonal queries

In this section, we present our results for Approximate Convex Hull in $\mathbb{R}^d$ via orthogonal emptiness queries. For non-adaptive queries (Section 3.1), the algorithm is very simple, while the lower bound proof is nontrivial and interesting. For adaptive queries (Section 3.2), the algorithm is more technical and the lower bound proof is simpler.

3.1 Non-adaptive orthogonal queries

We first present our algorithmic result with non-adaptive orthogonal emptiness queries. Let $P \subseteq [0,1]^d$ be an unknown set of points and $\mathcal{O}$ be the orthogonal emptiness oracle on P. For an axis-parallel box $\square$ in $\mathbb{R}^d$, let Query($\square$) denote the output of $\mathcal{O}$ when queried with $\square$, which is yes if $P \cap \square = \emptyset$ and is no if $P \cap \square \neq \emptyset$.

Algorithm 1 NonAdaptiveOrthogonal(q).

1: $r \leftarrow \lfloor q^{1/d} \rfloor$
2: $\Gamma \leftarrow$ Partition($[0,1]^d, r$)
3: $A_\square \leftarrow$ Query($\square$) for all $\square \in \Gamma$
4: $\Gamma_1 = \{\square \in \Gamma : A_\square = \text{no}\}$
5: $C^* \leftarrow \mathcal{CH}(\bigcup_{\square \in \Gamma_1} \square)$
6: **return** C^*

Our algorithm for approximating $\mathcal{CH}(P)$ is very simple (presented in Algorithm 1) and is similar to the algorithm of Czumaj and Sohler [16] for testing convex position. Let $r = \lfloor q^{1/d} \rfloor$. In line 2, the sub-routine Partition($[0,1]^d, r$) partitions $[0,1]^d$ evenly into r^d *cells* each of which is a hypercube of side-length $\frac{1}{r}$; let Γ be the set of the r^d cells. Then we query $\mathcal{O}$ with the boxes in Γ, and let $A_\square =$ Query($\square$) for $\square \in \Gamma$. Define Γ_1 as the set of cells $\square \in \Gamma$ with $A_\square =$ no (which are just the cells containing at least one point in P). Finally, the algorithm simply returns $C^* = \mathcal{CH}(\bigcup_{\square \in \Gamma_1} \square)$. Clearly, the number of queries to $\mathcal{O}$ is at most q, and they are non-adaptive as the algorithm performs them at the same time.

To bound the error of our algorithm, we need the following lemma about the volume of the Minkowski sum of a convex body and a ball; we include a proof in the full version. For two convex bodies X and Y in $\mathbb{R}^d$, denote by $X + Y = \{x + y : x \in X \text{ and } y \in Y\}$ their Minkowski sum.

▶ **Lemma 1.** *For constant d, let $C \subseteq [0,1]^d$ be a convex body and B_δ be the ball with radius $\delta \in [0,1]$ centered at the origin of $\mathbb{R}^d$. Then $\|(C + B_\delta)\backslash C\| = O(\delta)$ for all $\delta \in [0,1]$.*

By construction, we have $\mathcal{CH}(P) \subseteq C^*$. On the other hand, we observe that $C^* \subseteq \mathcal{CH}(P) + B$, where B is the ball in $\mathbb{R}^d$ centered at the origin with radius $\frac{d}{r}$. As C^* is a polytope, it suffices to show that every vertex of C^* lies in $\mathcal{CH}(P) + B$. A vertex z of C^* is a corner of some $\square \in \Gamma_1$. Since $A_\square = \mathsf{no}$, there exists a point $p \in P \cap \square$. The distance between z and p is at most $\frac{d}{r}$, as the side-length of $\square$ is $\frac{1}{r}$. Thus, $z \in p + B$ and $z \in \mathcal{CH}(P) + B$. It follows that $\|C^*\backslash\mathcal{CH}(P)\| \leq \|(\mathcal{CH}(P) + B)\backslash\mathcal{CH}(P)\|$. By Lemma 1, $\|(\mathcal{CH}(P) + B)\backslash\mathcal{CH}(P)\| = O(\frac{1}{r})$. So we have $\|C^*\backslash\mathcal{CH}(P)\| = O(\frac{1}{r}) = O(q^{-1/d})$.

▶ **Theorem 2.** *There exists an algorithm for* Approximate Convex Hull *in $\mathbb{R}^d$ that performs q non-adaptive orthogonal emptiness queries and has error $O(q^{-1/d})$.*

Interestingly, the above algorithm, while being very simple, is already the best one can hope. Specifically, we show that any deterministic algorithm for Approximate Convex hull with q non-adaptive orthogonal emptiness queries has error $\Omega(q^{-1/d})$.

Consider such an algorithm $\mathbf{A}$. Since the q queries $\mathbf{A}$ performs are non-adaptive, these queries are independent of the point set P (as well as the range-emptiness oracle). Let $\square_1, \ldots, \square_q$ be these queries, each of which is a box in $\mathbb{R}^d$. Without loss of generality, we may assume $\square_1, \ldots, \square_q \subseteq [0,1]^d$. Set $n = \lceil q^{1/d} \rceil + 1$ and $\delta = q^{-1/d}/c$ where c is a sufficiently large constant (which only depends on d). Define a sequence $H_0, H_1, \ldots, H_n$ of parallel hyperplanes in $\mathbb{R}^d$ where the equation of H_i is $x_1 + \cdots + x_d = 1 + \delta i$. For $i \in [n]$, we say a box $\square$ is *i-good* if $H_i \cap \square \neq \emptyset$ and $H_{i-1} \cap \square = \emptyset$. See figure 1.

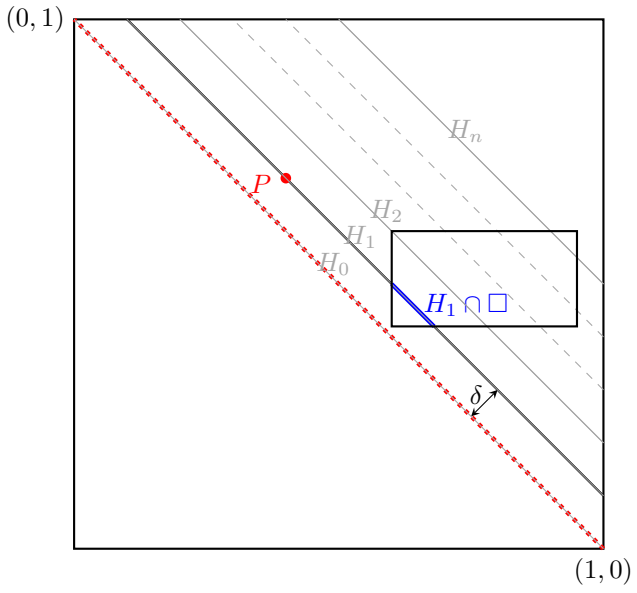


Figure 1 The lower bound construction in $d = 2$ dimensions for non-adaptive orthogonal queries with underlying points P (red) along the main diagonal H_0. The depicted query $\square$ is 1-good since its lower left corner lies between H_0 and H_1. The length of $H_1 \cap \square$ (blue) is $O(\delta^{d-1})$, so $\Omega(1/\delta^{d-1})$ 1-good queries are needed to cover H_1 in order to determine whether P contains a point on H_1. In total, $\Omega(n\delta^{d-1})$ i-good queries are needed to cover all $H_1, \cdots, H_n$.

▶ **Lemma 3.** *There exists $i \in [n]$ such that at most $\frac{q}{n}$ boxes in $\{\square_1, \ldots, \square_q\}$ are i-good.*

Proof. Observe that for different $i, j \in [n]$, no box in $\mathbb{R}^d$ can be both i-good and j-good. To see this, suppose $i < j$ without loss of generality. Let $\square$ be a j-good box. Then $H_j \cap \square \neq \emptyset$ and $H_{j-1} \cap \square = \emptyset$. So H_j and $\square$ are on the same side of H_{j-1}. If $i = j - 1$, then $H_i \cap \square = \emptyset$ and thus $\square$ is not i-good. Otherwise, $i \leq j - 2$. In this case, H_i and $\square$ are on the opposite sides of H_{j-1}, which implies $H_i \cap \square = \emptyset$ and $\square$ is not i-good. Therefore, any box in $\mathbb{R}^d$ can be i-good for at most one $i \in [n]$. Let t_i be the number of i-good boxes in $\{\square_1, \ldots, \square_q\}$. It follows that $\sum_{i=1}^{n} t_i \leq q$ and hence there exists $i \in [n]$ satisfying $t_i \leq q/n$. ◀

Let $i \in [n]$ such that at most $\frac{q}{n}$ boxes in $\{\square_1, \ldots, \square_q\}$ are i-good. Without loss of generality, assume $\square_1, \ldots, \square_t$ are i-good and $\square_{t+1}, \ldots, \square_q$ are not i-good, where $t \leq \frac{q}{n}$. Each $H_i \cap \square_j$ is a $(d-1)$-dimensional convex polytope, and let $\|H_i \cap \square_j\|'$ denote its $(d-1)$-dimensional volume.

▶ **Lemma 4.** *For all $j \in [t]$, $\|H_i \cap \square_j\|' = O(\delta^{d-1})$.*

Proof. Suppose $\square_j = [x_1^-, x_1^+] \times \cdots \times [x_d^-, x_d^+]$ and $\square_j' = [x_1^-, +\infty) \times \cdots \times [x_d^-, \infty)$. Since $\square_j$ is i-good, we have $H_i \cap \square_j \neq \emptyset$ and $H_{i-1} \cap \square_j = \emptyset$, which implies $1 + \delta i - \delta \leq x_1^- + \cdots + x_d^- \leq 1 + \delta i$. So the distance from the point $(x_1^-, \ldots, x_d^-) \in \mathbb{R}$ to H_i is $O(\delta)$. This implies $\|H_i \cap \square_j'\|' = O(\delta^{d-1})$, and thus $\|H_i \cap \square_j\|' = O(\delta^{d-1})$. ◀

▶ **Lemma 5.** $H_i \cap [0,1]^d \nsubseteq \bigcup_{j=1}^{t} \square_j$.

Proof. Equivalently, we prove that $H_i \cap [0,1]^d \nsubseteq \bigcup_{j=1}^{t} (H_i \cap \square_j)$. As $H_i \cap [0,1]^d$ and $H_i \cap \square_1, \ldots, H_i \cap \square_t$ are all $(d-1)$-dimensional convex polytopes, it suffices to show that $\|H_i \cap [0,1]^d\|' > \sum_{j=1}^{t} \|H_i \cap \square_j\|'$. Since $1 + \delta i = 1 + O(\frac{1}{c})$ and c is sufficiently large, $\|H_i \cap [0,1]^d\|'$ is lower bounded by a constant depending only on d. By Lemma 4, $\sum_{j=1}^{t} \|H_i \cap \square_j\|' = O(\delta^{d-1} t) = O(\frac{1}{c})$. Therefore, we have $\|H_i \cap [0,1]^d\|' > \sum_{j=1}^{t} \|H_i \cap \square_j\|'$, for sufficiently large c. ◀

Next, we construct two sets Z and Z' of points in $\mathbb{R}^d$. We include in Z the vertices of the polytope $H_{i-1} \cap [0,1]^d$. Furthermore, for each $j \in [q]$ such that $H_{i-1} \cap \square_j \neq \emptyset$, we include in Z a point $p_j \in H_{i-1} \cap \square_j$. Note that all points in Z lie on $H_{i-1} \cap [0,1]^d$ and we have $\mathcal{CH}(Z) = H_{i-1} \cap [0,1]^d$, so $\|\mathcal{CH}(Z)\| = 0$. To further construct Z', we pick a point $p \in (H_i \cap [0,1]^d) \backslash (\bigcup_{j=1}^{t} \square_j)$, which exists by Lemma 5. Then we define $Z' = Z \cup \{p\}$.

▶ **Lemma 6.** *For all $j \in [q]$, $Z \cap \square_j = \emptyset$ iff $Z' \cap \square_j = \emptyset$.*

Proof. If $p \notin \square_j$, then $Z \cap \square_j = Z' \cap \square_j$ and we are done. So assume $p \in \square_j$. Then $j \notin [t]$ by the choice of p. Thus, $\square_j$ is not i-good and we have either $H_i \cap \square_j = \emptyset$ or $H_{i-1} \cap \square_j \neq \emptyset$. But $H_i \cap \square_j \neq \emptyset$ since $p \in H_i \cap \square_j$. So we must have $H_{i-1} \cap \square_j \neq \emptyset$. By construction, we include in Z the point $p_j \in H_{i-1} \cap \square_j$, which implies $Z \cap \square_j \neq \emptyset$ and $Z' \cap \square_j \neq \emptyset$. ◀

The above lemma shows that the algorithm $\mathbf{A}$ cannot distinguish Z and Z', that is, it returns the same convex body when running on $P = Z$ and $P = Z'$.

▶ **Lemma 7.** $\|\mathcal{CH}(Z') \backslash \mathcal{CH}(Z)\| = \Omega(\delta)$.

Let C^* be the output of $\mathbf{A}$ when run on $P = Z$ or $P = Z'$. We have $\|\mathcal{CH}(Z') \backslash \mathcal{CH}(Z)\| \leq \|C^* \oplus \mathcal{CH}(Z)\| + \|C^* \oplus \mathcal{CH}(Z')\|$. Therefore, the above lemma implies either $\|C^* \oplus \mathcal{CH}(Z)\| = \Omega(\delta)$ or $\|C^* \oplus \mathcal{CH}(Z')\| = \Omega(\delta)$. As $\delta = \Theta(q^{-1/d})$, the error of $\mathbf{A}$ is $\Omega(q^{-1/d})$.

▶ **Theorem 8.** *Any deterministic algorithm for* Approximate Convex Hull *in $\mathbb{R}^d$ that performs q non-adaptive orthogonal emptiness queries has error $\Omega(q^{-1/d})$.*

3.2 Adaptive orthogonal queries

We now consider adaptive orthogonal emptiness queries, and show how to approximate $\mathcal{CH}(P)$ with error $O(q^{-1/(d-1)})$ via $O(q)$ queries. We shall define 2^d supersets of $\mathcal{CH}(P)$, one-to-one corresponding to the vectors in $\{-1,1\}^d$. Our algorithm independently approximates these supersets and then merges them together. For each $\vec{v} = (v_1, \ldots, v_d) \in \{-1,1\}^d$, let O_v be the orthant $\{(x_1, \ldots, x_d) : x_i v_i \geq 0 \text{ for all } i \in [d]\}$. We define $\mathcal{CH}_{\vec{v}}(P) = (\mathcal{CH}(P) + O_{-\vec{v}}) \cap [0,1]^d$. The following observation relates $\mathcal{CH}(P)$ with the supersets $\mathcal{CH}_{\vec{v}}(P)$ (proof in full version).

▶ **Lemma 9.** $\mathcal{CH}(P) = \bigcap_{\vec{v} \in \{-1,1\}^d} \mathcal{CH}_{\vec{v}}(P)$.

Algorithm 2 presents how to approximate $\mathcal{CH}_{\vec{v}}(P)$ for each $\vec{v} \in \{-1,1\}^d$. It essentially "sandwiches" the hull between two convex boundaries, refining the estimate at each iteration. Figure 2 illustrates one such refinement step. The subroutine SUBDIVIDE(R) evenly partitions the hypercube R into 2^d smaller hypercubes (each of which has side-length half of the side-length of R) and returns the set of these 2^d hypercubes. For a hypercube R and a vector $\vec{v} \in \{-1,1\}^d$, we define $\mathsf{cor}(R, \vec{v}) = \arg\max_{p \in R} \langle p, \vec{v} \rangle$ as the corner of R in the direction $\vec{v}$. The algorithm starts with a set $\mathcal{R}$ of hypercubes which initially consists of only $[0,1]^d$, and runs in $\lceil \frac{\log q}{d-1} \rceil$ rounds. In each round, we perform emptiness queries for the hypercubes in $\mathcal{R}$, and let $\mathcal{R}' \subseteq \mathcal{R}$ consist of the nonempty ones. Then define $U = \mathcal{CH}_{\vec{v}}(\{\mathsf{cor}(R, \vec{v}) : R \in \mathcal{R}'\})$ and $L' = \mathcal{CH}_{\vec{v}}(\{\mathsf{cor}(R, -\vec{v}) : R \in \mathcal{R}'\})$. Let $L = L' \setminus \partial L'$ be the interior of L'. This is simply to ensure that $\partial \mathcal{CH}_{\vec{v}}(P)$ is contained in $U \setminus L$. We then take all hypercubes in $\mathcal{R}$ that intersect $U \setminus L$, partition them using the subroutine SUBDIVIDE, and let the new $\mathcal{R}$ be the set of the resulting smaller hypercubes. (Technically, we only need to consider the nonempty queries in $\mathcal{R}'$, but this is equivalent in the worst case, and this way nicely ensures that our queries continue to cover all of $\partial CH_{\vec{v}}(P)$). Finally, we return the convex body U in the last iteration as an approximation of $\mathcal{CH}_{\vec{v}}(P)$.

Algorithm 2 ADAPTIVEORTHOGONAL($q, \vec{v}$).

```
1: R ← {[0,1]^d}
2: for i = 1 to ⌈log q / (d−1)⌉ do
3:     R' ← {R ∈ R : QUERY(R) = no}
4:     U ← CH_v({cor(R, v) : R ∈ R'})
5:     L' ← CH_v({cor(R, −v) : R ∈ R'})
6:     L ← L' \ ∂L'
7:     R'' ← {R ∈ R : R ∩ (U \ L) ≠ ∅}
8:     R ← ⋃_{R∈R''} SUBDIVIDE(R)
9: return U
```

At a high level, the algorithm maintains an upper and lower bound (the sets U and L respectively) on the unknown convex body $\mathcal{CH}_{\vec{v}}(P)$. At each iteration of the loop, we reduce the error volume $\|(U \setminus \mathcal{CH}_{\vec{v}}(P)) \cap [0,1]^d\|$ by half. Let $t = \lceil \frac{\log q}{d-1} \rceil$ be the number of iterations. For $i \in [t]$, denote by $\mathcal{R}_i$ the set $\mathcal{R}$ at the end of the i-th iteration of the loop, and likewise for L_i, U_i, $\mathcal{R}'_i$, $\mathcal{R}''_i$. Without loss of generality, we assume that $\vec{v} = (1, \ldots, 1)$. The following two lemmas are used to bound the error and number of queries; their proofs are included in the full version.

▶ **Lemma 10.** *For all $i \in [t]$, $L_i \subset \mathcal{CH}_{\vec{v}}(P) \subseteq U_i$ and $\partial \mathcal{CH}_{\vec{v}}(P) \subseteq \bigcup_{R \in \mathcal{R}_i} R$.*

▶ **Lemma 11.** *The number of queries at the i-th iteration is at most $4^d d! \cdot (2^i + 1)^{d-1}$.*

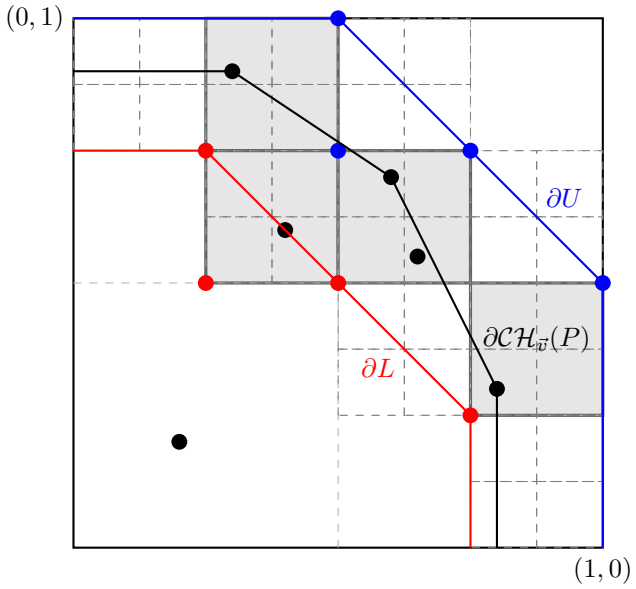


Figure 2 One iteration of Algorithm 2 for $d = 2, \vec{v} = (1, 1)$. The four queries in $\mathcal{R}'$ are filled in gray. The upper right corners of these queries are used to compute the set U with boundary shown in blue, while the lower left corners produce L in red. The boundary of $\mathcal{CH}_{\vec{v}}(P)$ lies in $U \setminus L$. The smallest dashed queries form the set $\mathcal{R}$ for the next iteration; their union covers $U \setminus L$.

As d is a constant, Lemma 11 implies the number of queries in the i-th iteration is $O((2^i+1)^{d-1})$. So the total number of queries is bounded by $\sum_{i=1}^{t}(2^i+1)^{d-1} = O(q)$. Since each query box in the i-th iteration has volume 2^{-id}, we have

$$\left\| \bigcup_{R \in \mathcal{R}_t} R \right\| \leq 2^{-td}|\mathcal{R}_t| = O(2^{-d \log q/(d-1)} q) = O(q^{-1/(d-1)}).$$

By Lemma 10, this further implies $\|(U_t \backslash \mathcal{CH}_{\vec{v}}(P)) \cap [0,1]^d\| = O(q^{-1/(d-1)})$, simply because $(U_t \backslash \mathcal{CH}_{\vec{v}}(P)) \cap [0,1]^d \subseteq (U_t \backslash L_t) \cap [0,1]^d \subseteq \bigcup_{R \in \mathcal{R}_t} R$.

We run Algorithm 2 for all $\vec{v} \in \{-1, 1\}^d$ and let $U_{\vec{v}}$ be the output for $\vec{v}$. Finally, we return $C^* = (\bigcap_{\vec{v} \in \{-1,1\}^d} U_{\vec{v}}) \cap [0,1]^d$ as the approximation of $\mathcal{CH}(P)$. The total number of queries is still $O(q)$. To bound the error $\|C^* \oplus \mathcal{CH}(P)\|$, we first observe $\mathcal{CH}(P) \subseteq C^*$. Indeed, since $\mathcal{CH}_{\vec{v}}(P) \subseteq U_{\vec{v}}$ for all $\vec{v} \in \{-1, 1\}^d$, we have $\mathcal{CH}(P) \subseteq \bigcap_{\vec{v} \in \{-1,1\}^d} U_{\vec{v}}$ by Lemma 9, which implies $\mathcal{CH}(P) \subseteq C^*$ because $P \subseteq [0,1]^d$. Again, by Lemma 9, we have

$$\|C^* \backslash \mathcal{CH}(P)\| \leq \sum_{\vec{v} \in \{-1,1\}^d} \|(U_{\vec{v}} \backslash \mathcal{CH}_{\vec{v}}(P)) \cap [0,1]^d\| = O(q^{-1/(d-1)}).$$

Therefore, the error of our algorithm is $O(q^{-1/(d-1)})$.

▶ **Theorem 12.** *There exists an algorithm for* Approximate Convex Hull *in* $\mathbb{R}^d$ *that performs* $O(q)$ *adaptive orthogonal emptiness queries and has error* $O(q^{-1/(d-1)})$.

We complement this result by showing that any deterministic algorithm with q adaptive orthogonal queries has error $\Omega(q^{-1/(d-1)})$. The argument is similar to and simpler than that of Theorem 8, so we include the precise details in the full version. The key difference is that Lemma 3 no longer holds for adaptive queries, but we can still apply the same argument for a single pair of hyperplanes and only lose a factor $n = q^{1/d}$ in the resulting error bound.

▶ **Theorem 13.** *Any deterministic algorithm for* Approximate Convex Hull *in* $\mathbb{R}^d$ *that performs* q *adaptive orthogonal emptiness queries has error* $\Omega(q^{-1/(d-1)})$.

4 Approximating convex hulls via halfplane queries

In this section, we present our results for Approximate Convex Hull in $\mathbb{R}^2$ via halfplane emptiness queries. Instead of working on halfplane emptiness queries directly, it is more convenient to consider *extreme halfplane queries*, which we formalize below. For two parallel lines ℓ and ℓ', let $\mathsf{dist}(\ell, \ell')$ denote the distance between them.

▶ **Definition 14** (extreme halfplane oracle)**.** *An* ***extreme halfplane oracle*** *on P takes a unit vector $\vec{v} \in \mathbb{S}^1$ as input and returns a halfplane H with normal vector $\vec{v}$ such that $P \subseteq H$. We say the oracle is* ***δ-accurate*** *for a number $\delta \geq 0$ if for every $\vec{v} \in \mathbb{S}^1$, the halfplane H returned by the oracle for query $\vec{v}$ satisfies the condition $\mathsf{dist}(\partial H, \partial H^*) \leq \delta$, where H^* is the minimal halfplane with normal vector $\vec{v}$ satisfying $P \subseteq H^*$.*

It is easy to see that a 0-accurate extreme halfplane oracle is stronger than a halfplane emptiness oracle. On the other hand, one can simulate an extreme halfplane query with accuracy δ via $O(1/\delta)$ evenly spaced non-adaptive halfplane emptiness queries. The query number can be reduced to $O(\log(1/\delta))$ via binary search in the adaptive model. The proofs of the following two lemmas are included in the full version.

▶ **Lemma 15.** *Given access to a halfplane emptiness oracle $\mathcal{O}$ on an unknown set P of points in the plane, one can build a δ-accurate extreme halfplane oracle on P which performs $O(\frac{1}{\delta})$ non-adaptive queries to $\mathcal{O}$ or performs $O(\log \frac{1}{\delta})$ adaptive queries to $\mathcal{O}$.*

We will also need the following lemma to argue that the error introduced by querying a δ-accurate extreme halfplane oracle, as opposed to a perfectly accurate one, is only $O(\delta)$.

▶ **Lemma 16.** *Let $H_1, \ldots, H_r$ be halfplanes, $C = \bigcap_{i=1}^r H_i$, and $\delta \in [0,1]$. Suppose $C \subseteq [0,1]^2$. For each $i \in [r]$, let H_i' be a halfplane parallel to H_i such that $C \subseteq H_i$ and $\mathsf{dist}(\partial H_i', \partial H_i) \leq \delta$. Define $C' = \bigcap_{i=1}^r H_i'$. Then $\|(C' \cap [0,1]^2) \oplus C\| = O(\delta)$.*

In what follows, we discuss the results for non-adaptive halfplane queries and adaptive halfplane queries individually. Same as before, for non-adaptive queries (Section 4.1), the algorithm is simple and the lower bound proof is nontrivial, while for adaptive queries (Section 4.2), the algorithm is more technical and the lower bound proof is simpler.

4.1 Non-adaptive halfplane queries

We first present our algorithmic result with non-adaptive halfplane emptiness queries. Let $P \subseteq [0,1]^2$ be a (unknown) set of points. We shall show how to approximate $\mathcal{CH}(P)$ with error $O(\frac{1}{q} + \delta)$ via q non-adaptive queries to a δ-accurate extreme halfplane oracle $\mathcal{E}$ on P. Setting $\delta = \frac{1}{q}$ and applying Lemma 15 yields an algorithm with error $O(1/q)$ that makes q^2 non-adaptive halfplane emptiness queries. Equivalently, q queries achieve error $O(1\sqrt{q})$.

Algorithm 3 NonAdaptiveHalfplane(q).

1: $V \leftarrow \{(\sin \frac{2i\pi}{q}, \cos \frac{2i\pi}{q}) : i \in [q]\}$
2: $H_{\vec{v}} \leftarrow \textsc{Query}(\vec{v})$ for all $\vec{v} \in V$
3: $C^* \leftarrow \bigcap_{\vec{v} \in V} H_{\vec{v}}$
4: **return** $C^* \cap [0,1]^2$

For a unit vector $\vec{v} \in \mathbb{S}^1$, let $\textsc{Query}(\vec{v})$ denote the output of $\mathcal{E}$ when queried with $\vec{v}$. Our algorithm for approximating $\mathcal{CH}(P)$ is very simple, and is presented in Algorithm 3. Let $V = \{(\sin \frac{2i\pi}{q}, \cos \frac{2i\pi}{q}) : i \in [q]\}$ be q unit vectors uniformly picked on $\mathbb{S}^1$. We query $\mathcal{E}$

with the vectors in V, and let $H_{\vec{v}} = \textsc{Query}(\vec{v})$ for all $\vec{v} \in V$. After that, we simply return $C^* \cap [0,1]^2$ where $C^* = \bigcap_{\vec{v} \in V} H_{\vec{v}}$. Clearly, the number of queries to $\mathcal{E}$ is q, and they are non-adaptive as the algorithm performs them at the same time. We bound the error to $O(1/\sqrt{q})$ in the full version.

▶ **Theorem 17.** *There exists an algorithm for* Approximate Convex Hull *in* $\mathbb{R}^2$ *that performs* q *non-adaptive halfplane emptiness queries and has error* $O(1/\sqrt{q})$.

Next, we show that any deterministic algorithm that performs q non-adaptive halfplane queries must have error $\Omega(1/\sqrt{q})$. Consider an algorithm $\mathbf{A}$. As the q queries performed by $\mathbf{A}$ are non-adaptive, they are independent of the point set P. Let $H_1, \ldots, H_q$ be these queries, each of which is a halfplane in $\mathbb{R}^2$. For $x, y \in (0,1)$, denote by $\ell_{x,y}$ the line that goes through the points $(x,0)$ and $(0,y)$. Also, for $X, Y \subseteq (0,1)$, define $\mathcal{L}_{X,Y} = \{\ell_{x,y} : x \in X \text{ and } y \in Y\}$. Fix a parameter $\delta = \frac{1}{10\sqrt{q}}$, and let $A = \{i\delta : i \in \mathbb{Z}\} \cap [\frac{1}{3}, \frac{2}{3}]$.

▶ **Lemma 18.** *There exists* $a, b \in A$ *such that* $\partial H_i \notin \mathcal{L}_{[a,a+\delta),[b,b+\delta)}$ *for all* $i \in [q]$.

Proof. Observe that for different $a, a' \in A$, the intervals $[a, a+\delta)$ and $[a', a'+\delta)$ are disjoint. Therefore, $\mathcal{L}_{[a,a+\delta),[b,b+\delta)}$ and $\mathcal{L}_{[a',a'+\delta),[b',b'+\delta)}$ are disjoint, for $(a,b), (a',b') \in A \times A$ with $(a,b) \neq (a',b')$. It follows that for each $i \in [q]$, there exists at most one pair $(a,b) \in A \times A$ such that $\partial H_i \in \mathcal{L}_{[a,a+\delta),[b,b+\delta)}$; we charge i to (a,b) if this is the case. By construction, $|A| \geq 3\sqrt{q}$ and thus $|A \times A| \geq 9q > q$. As such, there exists a pair $(a,b) \in A \times A$ such that no number in $[q]$ is charged to (a,b). We then have $\partial H_i \notin \mathcal{L}_{[a,a+\delta),[b,b+\delta)}$ for all $i \in [q]$. ◀

Let $a, b \in A$ be as in the above lemma, and $p^* = (\frac{a}{2} + \frac{\delta}{10}, \frac{b}{2} + \frac{\delta}{10})$. We define two sets $Z = \{(a,0), (0,b)\}$ and $Z' = \{(a,0), (0,b), p^*\}$; see figure 3.

▶ **Lemma 19.** *The point* p^* *is below the lines* $\ell_{a+\delta,b}$ *and* $\ell_{a,b+\delta}$.

Proof. Without loss of generality, we only need to show p^* is below $\ell_{a+\delta,b}$. The equation of $\ell_{a+\delta,b}$ is $bx + (a+\delta)y - b(a+\delta) = 0$. Since $a, b \in A$, we have $a, b \in [\frac{1}{3}, \frac{2}{3}]$. So we have

$$b\left(\frac{a}{2} + \frac{\delta}{10}\right) + (a+\delta)\left(\frac{b}{2} + \frac{\delta}{10}\right) - b(a+\delta) = \left(\frac{a+b+\delta}{10} - \frac{b}{2}\right)\delta < 0,$$

which implies that that p^* is below $\ell_{a+\delta,b}$. ◀

▶ **Corollary 20.** *For all* $i \in [q]$, $Z \cap H_i = \emptyset$ *iff* $Z' \cap H_i = \emptyset$.

Proof. If $p^* \notin H_i$, then $Z \cap H_i = Z' \cap H_i$ and we are done. So suppose $p^* \in H_i$. We claim that either $(a,0) \in H_i$ or $(0,b) \in H_i$. It suffices to show that at least one of $(a,0)$ and $(0,b)$ lies on the same side of ∂H_i as p^*. Assume both $(a,0)$ and $(0,b)$ are on the opposite side of ∂H_i from p^*. Then p^* and the segment $\ell_{a,b} \cap [0,1]^2$ must lie on the opposite side of ∂H_i. But by our construction, p^* is sufficiently close to $\ell_{a,b} \cap [0,1]^2$, which forces $\partial H_i = \ell_{x,y}$ for some $x \in [a,1]$ and $y \in [b,1]$. By Lemma 18, we cannot have $x \in [a, a+\delta)$ and $y \in [b, b+\delta)$ at the same time. So either $x \geq a+\delta$ or $y \geq b+\delta$. Without loss of generality, assume $x \geq a+\delta$. By Lemma 19, p^* is below $\ell_{a+\delta,b}$. Since $x \geq a+\delta$ and $y \geq b$, p^* is below ∂H_i as well. But both $(a,0)$ and $(0,b)$ are also below ∂H_i, contradicting our assumption. As such, we have either $(a,0) \in H_i$ or $(0,b) \in H_i$. It follows that $Z \cap H_i \neq \emptyset$ and $Z' \cap H_i \neq \emptyset$. ◀

The rest of the proof is almost the same as that of Theorem 8. The above lemma shows that the algorithm $\mathbf{A}$ cannot distinguish Z and Z', that is, it returns the same convex polygon when running on $P = Z$ and $P = Z'$. We only need to observe the following simple fact.

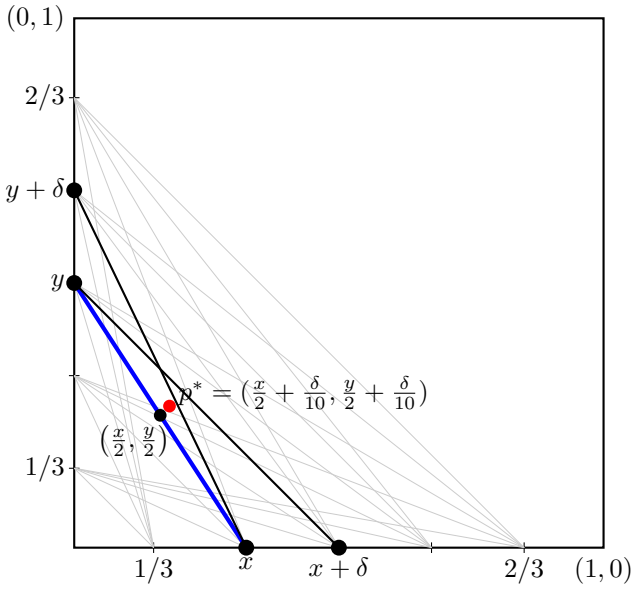


Figure 3 Lower bound construction for non-adaptive halfplane queries. The point p^* (red) is offset from the midpoint $(x/2, y/2)$ of $l_{x,y}$ to lie sufficiently far above $l_{x,y}$ (blue) but below both $l_{x+\delta,y}$ and $l_{x,y+\delta}$. Only a segment with both x-intercept in the interval $[x, x+\delta]$ and y-intercept in the interval $[y, y+\delta]$ can separate both $(x, 0), (0, y)$ from p^*. Doing so for all $\Theta(1/\delta^2) > q$ paired choices of x, y (gray lines) necessitates as many queries. Range [1/3, 2/3] enlarged for clarity.

▶ **Lemma 21.** $\|\mathcal{CH}(Z')\backslash\mathcal{CH}(Z)\| = \Omega(\delta)$.

Let C^* be the output of $\mathbf{A}$ when running on $P = Z$ or $P = Z'$. We have $\|\mathcal{CH}(Z')\backslash\mathcal{CH}(Z)\| \leq \|C^* \oplus \mathcal{CH}(Z)\| + \|C^* \oplus \mathcal{CH}(Z')\|$. Thus, the above lemma implies either $\|C^* \oplus \mathcal{CH}(Z)\| = \Omega(\delta)$ or $\|C^* \oplus \mathcal{CH}(Z')\| = \Omega(\delta)$. As $\delta = \Theta(1/\sqrt{q})$, the error of $\mathbf{A}$ is $\Omega(1/\sqrt{q})$.

▶ **Theorem 22.** *Any deterministic algorithm for* Approximate Convex Hull *in* $\mathbb{R}^2$ *that performs* q *non-adaptive halfplane emptiness queries incurs error* $\Omega(1/\sqrt{q})$.

4.2 Adaptive halfplane queries

We first present our algorithm with adaptive halfplane emptiness queries. Let $P \subseteq [0,1]^2$ be a (unknown) set of points. We shall show how to approximate $\mathcal{CH}(P)$ with error $O(\frac{1}{q^2} + q^2\delta)$ via $O(q \log q)$ adaptive queries to a δ-accurate extreme halfplane oracle $\mathcal{E}$ on P. Combining this with Lemma 15 will give us the desired result.

Same as before, for a unit vector $\vec{v} \in \mathbb{S}^1$, let Query$(\vec{v})$ denote the output of $\mathcal{E}$ when queried with $\vec{v}$. Our algorithm for approximating $\mathcal{CH}(P)$ is presented in Algorithm 4. We begin with a set V of unit vectors which is initially $\{(1,0),(0,1),(-1,0),(0,-1)\}$. We query $\mathcal{E}$ with the vectors in V and let $H_{\vec{v}} = \text{Query}(\vec{v})$. Then we iteratively add more vectors to V and query $\mathcal{E}$ with these vectors (the repeat-until loop in the algorithm). In each round, we first compute $C^* = \bigcap_{\vec{v}\in V} H_{\vec{v}}$ for the current V (line 4). Then line 5 sorts the vectors in the current V in clockwise order; let $\vec{v}_1, \ldots, \vec{v}_n$ be the sorted sequence. Let $\vec{v}_0 = \vec{v}_n$ and $\sigma_i = C^* \cap \partial H_{\vec{v}_i}$, which is the edge of C^* corresponding to the halfplane $H_{\vec{v}_i}$. We consider each pair $(\vec{v}_{i-1}, \vec{v}_i)$ of adjacent vectors satisfying $\mathsf{ang}(\vec{v}_{i-1}, \vec{v}_i) \cdot |\sigma_i| > \frac{1}{q^2}$, where $\mathsf{ang}(\vec{v}_{i-1}, \vec{v}_i)$ denotes the magnitude of the angle from $\vec{v}_{i-1}$ to $\vec{v}_i$. For each such pair $(\vec{v}_{i-1}, \vec{v}_i)$, we add a

new unit vector $\vec{v}$ to V, which is the bisector of $\vec{v}_{i-1}$ and $\vec{v}_i$, and query $\mathcal{E}$ with $\vec{v}$ to obtain the halfplane $H_{\vec{v}} = \textsc{Query}(\vec{v})$. After this, we proceed to the next round. The procedure terminates when no more vectors are added to V in a round. At the end, the algorithm returns the convex polygon $C^* \cap [0,1]^2$ as its output.

Algorithm 4 AdaptiveHalfplane(q).

$V \leftarrow \{(1,0),(0,1),(-1,0),(0,-1)\}$
$H_{\vec{v}} \leftarrow \textsc{Query}(\vec{v})$ for all $\vec{v} \in V$
repeat
  $C^* \leftarrow \bigcap_{\vec{v}\in V} H_{\vec{v}}$
  $(\vec{v}_1,\ldots,\vec{v}_n) \leftarrow \textsc{Sort}(V)$
  $\vec{v}_0 \leftarrow \vec{v}_n$
  $\sigma_i \leftarrow C^* \cap \partial H_{\vec{v}_i}$ for $i \in [n]_0$
  $I \leftarrow \{i \in [n] : \mathsf{ang}(\vec{v}_{i-1},\vec{v}_i) \cdot |\sigma_i| > \frac{1}{q^2}\}$
  for every $i \in I$ **do**
    $\vec{v} \leftarrow (\vec{v}_{i-1}+\vec{v}_i)/\|\vec{v}_{i-1}+\vec{v}_i\|_2$
    $V \leftarrow V \cup \{\vec{v}\}$
    $H_{\vec{v}} \leftarrow \textsc{Query}(\vec{v})$
until $I = \emptyset$
return $C^* \cap [0,1]^2$

Bounding the number of queries. Clearly, the number of queries we made in Algorithm 4 is just equal to the number of vectors in V at the end of the algorithm. Suppose the repeat-until loop in Algorithm 4 has t iterations. For $i \in [t]$, let V_i be the set V at the beginning of the i-th iteration and $C_i^* = \bigcap_{\vec{v}\in V_i} H_{\vec{v}}$ which is just the convex polygon C^* in the i-th iteration. Note that V_t is just the set V at the end of the algorithm, as the algorithm does not add new vectors to V in the last iteration. For convenience, we write $V_0 = \emptyset$. Denote by $\mathcal{A}(V_i)$ the set of angles between adjacent vectors in V_i. We first bound $|V_i \backslash V_{i-1}|$ for $i \in [t]$. The proofs for this subsection are included in the full version.

▶ **Lemma 23.** *$|V_i \backslash V_{i-1}| = O(q)$ for all $i \in [t]$.*

The above lemma directly implies $|V_t| = O(qt)$. Thus, it suffices to bound t. To this end, we establish the following properties for the angles in each $\mathcal{A}(V_i)$.

▶ **Lemma 24.** *For all $i \in [t]$ and $\alpha \in \mathcal{A}(V_i)$, we have $|\alpha| \geq \frac{1}{12q^2}$.*

▶ **Lemma 25.** *Let $i \in [t]$. We have $|\alpha| = \frac{\pi}{2^i}$ for all $\alpha \in \mathcal{A}(V_i) \backslash \mathcal{A}(V_{i+1})$.*

Lemma 24 and Lemma 25 together imply that $t = O(\log q)$. Indeed, $\mathcal{A}(V_i) \neq \mathcal{A}(V_{i+1})$ for all $i \in [t-1]$, for otherwise the algorithm terminates after the i-th iteration. Thus there exists $\alpha \in \mathcal{A}(V_{t-1}) \backslash \mathcal{A}(V_t)$. We have $\frac{\pi}{2^{t-1}} = |\alpha| \geq \frac{1}{q^2}$ by Lemma 24 and Lemma 25, which implies $t = O(\log q)$. It follows that $|V_t| = O(qt) = O(q \log q)$.

Bounding the error. Next, we show that the output $C^* \cap [0,1]^2$ of Algorithm 4 satisfies $\|(C^* \cap [0,1]^2) \oplus \mathcal{CH}(P)\| = O(\frac{1}{q^2} + \delta)$. Let $V = \{\vec{v}_1,\ldots,\vec{v}_n\}$ be the set of vectors at the end of Algorithm 4, where $\vec{v}_1,\ldots,\vec{v}_n$ are sorted in clockwise order. Set $\vec{v}_0 = \vec{v}_n$. As in the algorithm, for each $i \in [n]_0$, let $H_{\vec{v}_i} = \textsc{Query}(\vec{v}_i)$ and $\sigma_i = C^* \cap \partial H_{\vec{v}_i}$. Then $C^* = \bigcap_{i=1}^n H_{\vec{v}_i}$. We have $|\mathsf{ang}(\vec{v}_{i-1},\vec{v}_i)| \cdot (|\sigma_{i-1}| + |\sigma_i|) > \frac{1}{q^2}$ for all $i \in [n]_0$, for otherwise the repeat-until loop

in Algorithm 4 cannot terminate. Define $L_{\vec{v}_i}$ as the minimal halfplane with normal vector $\vec{v}_i$ that contains P and $C = \bigcap_{i=1}^{n} L_{\vec{v}_i}$. We have $C \subseteq C^*$, since $L_{\vec{v}_i} \subseteq H_{\vec{v}_i}$ for all $i \in [n]$. Also, we have $C \subseteq [0,1]^2$, since V consists of the four vectors $(1,0),(0,1),(-1,0),(0,-1)$, which guarantees that C is contained in the axis-parallel bounding box of P and thus contained in $[0,1]^2$. Now $\mathcal{CH}(P) \subseteq C \subseteq C^* \cap [0,1]^2$. By Lemma 16, $\|(C^* \cap [0,1]^2) \backslash C\| = O(\delta)$. So it suffices to show $\|C \backslash \mathcal{CH}(P)\| = O(\frac{1}{q^2} + \delta)$. Define $\mu_i = C \cap \partial L_{\vec{v}_i}$ for $i \in [n]$ and $\mu_0 = \mu_n$.

▶ **Lemma 26.** *$|\mu_i| \leq |\sigma_i| + O(q^2\delta)$ for all $i \in [n]$.*

Proof. Let a and a' be the endpoints of μ_i, where a (resp., a') is on the counterclockwise (resp., clockwise) side of μ_i with respect to C. We first consider the general case $\sigma_i \neq \emptyset$ and then discuss the special case $\sigma_i = \emptyset$. If $\sigma_i \neq \emptyset$, let b and b' be the endpoints of σ_i, where b (resp., b') is on the counterclockwise (resp., clockwise) side of σ_i with respect to C^*. As $\sigma_i \subseteq \partial H_{\vec{v}_i}$ and $\mu_i \subseteq \partial L_{\vec{v}_i}$, the 4-gon $aa'b'b$ is a trapezoid with parallel edges μ_i and σ_i. Therefore, we have the equation

$$|\mu_i| = |\sigma_i| + \frac{h}{\tan \angle baa'} + \frac{h}{\tan \angle b'a'a}.$$

where h is the height of this trapezoid. We have $h \leq \delta$ since $\mathsf{dist}(\partial H_{\vec{v}_i}, \partial L_{\vec{v}_i}) \leq \delta$. Furthermore, we observe that $|\angle baa'| \geq \mathsf{ang}(\vec{v}_{i-1}, \vec{v}_i)$ and $|\angle b'a'a| \geq \mathsf{ang}(\vec{v}_i, \vec{v}_{i+1})$. Indeed, $\angle baa' = \pi - \angle b'ba$. Since $a \in C^*$, the angle $\angle b'ba$ is smaller than or equal to the angle of C^* at the vertex b, where the latter is just $\pi - \mathsf{ang}(\vec{v}_{i-1}, \vec{v}_i)$. Thus,

$$|\angle baa'| = \pi - |\angle b'ba| \geq \pi - (\pi - \mathsf{ang}(\vec{v}_{i-1}, \vec{v}_i)) = \mathsf{ang}(\vec{v}_{i-1}, \vec{v}_i).$$

For the same reason, $|\angle b'a'a| \geq \mathsf{ang}(\vec{v}_i, \vec{v}_{i+1})$. By Lemma 24, it follows that $|\angle baa'| \geq \frac{1}{12q^2}$ and $|\angle b'a'a| \geq \frac{1}{12q^2}$. As such, $|\mu_i| = |\sigma_i| + O(q^2\delta)$.

If $\sigma_i = \emptyset$, then C^* is contained in the interior of $H_{\vec{v}_i}$. In this case, C^* has a unique vertex that is closest to $\partial H_{\vec{v}_i}$. Let both b and b' be this vertex. The same argument applies. ◀

The definition of each $L_{\vec{v}_i}$ guarantees that there exists a point $p_i \in P$ satisfying $p_i \in \partial L_{\vec{v}_i}$. As $p_i \in P \subseteq C$, we have $p_i \in \mu_i$. Let $P' = \{p_1, \ldots, p_n\}$. We have $P' \subseteq P$ and thus $\|C \backslash \mathcal{CH}(P')\| \geq \|C \backslash \mathcal{CH}(P)\|$. We shall show $\|C \backslash \mathcal{CH}(P')\| = O(\frac{1}{q^2} + \delta)$, which bounds $\|C \backslash \mathcal{CH}(P)\|$ as well. For each $i \in [n]$, let a_i be the intersection point of μ_{i-1} and μ_i, and τ_i be the segment with endpoints p_{i-1} and p_i. Note that $\|C \backslash \mathcal{CH}(P')\| = \sum_{i=1}^{n} \|\triangle p_{i-1}p_ia_i\|$.

▶ **Lemma 27.** *$\|\triangle p_{i-1}p_ia_i\| = O(|\tau_i| \cdot (\frac{1}{q^2} + q^2\delta))$ for all $i \in [n]$.*

Proof. By construction, the distance between p_{i-1} and p_i is just $|\tau_i|$. So we only need to show that the height h of $\triangle p_{i-1}p_ia_i$ with respect to the edge $p_{i-1}p_i$ is bounded by $O(\frac{1}{q^2} + q^2\delta)$. Let $\theta = \angle a_ip_ip_{i-1}$. As the distance between a_i and p_i is at most $|\mu_i|$, we have

$$h \leq |\mu_i| \cdot \sin|\theta| \leq |\mu_i| \cdot \theta \leq \mathsf{ang}(\vec{v}_{i-1}, \vec{v}_i) \cdot |\mu_i|.$$

By Lemma 26, this implies $h = \mathsf{ang}(\vec{v}_{i-1}, \vec{v}_i) \cdot |\sigma_i| + O(q^2\delta)$. Since $\mathsf{ang}(\vec{v}_{i-1}, \vec{v}_i) \cdot |\sigma_i| \leq \frac{1}{q^2}$, we have $h = O(\frac{1}{q^2} + q^2\delta)$ and thus $\|\triangle p_{i-1}p_ia_i\| = O(|\tau_i| \cdot (\frac{1}{q^2} + q^2\delta))$. ◀

The above lemma directly implies

$$\|C \backslash \mathcal{CH}(P')\| = \sum_{i=1}^{n} \|\triangle p_{i-1}p_ia_i\| = O((\sum_{i=1}^{n} |\tau_i|) \cdot (\frac{1}{q^2} + q^2\delta)).$$

As $\tau_1, \ldots, \tau_n$ are just the edge of the convex polygon $\mathcal{CH}(P')$, we have $\sum_{i=1}^{n} |\tau_i| = O(1)$ and thus $\|C \backslash \mathcal{CH}(P')\| = O(\frac{1}{q^2} + q^2\delta)$. As Lemma 16 implies $\|(C^* \cap [0,1]^2) \backslash C\| = O(\delta)$, we have $\|(C^* \cap [0,1]^2) \oplus \mathcal{CH}(P)\| \leq \|(C^* \cap [0,1]^2) \backslash \mathcal{CH}(P')\| = O(\frac{1}{q^2} + q^2\delta)$. Setting $\delta = \frac{1}{q^4}$, we see that one can approximate $\mathcal{CH}(P)$ with error $O(\frac{1}{q^2})$ via $O(q \log q)$ queries to a $\frac{1}{q^4}$-accurate extreme halfplane oracle for P.

Lemma 15 shows that a query to a $\frac{1}{q^4}$-accurate extreme halfplane oracle can be simulated with $O(\log q)$ adaptive halfplane emptiness queries. Therefore, one can approximate $\mathcal{CH}(P)$ with error $O(\frac{1}{q^2})$ via $O(q \log^2 q)$ adaptive halfplane emptiness queries. Equivalently, if we are allowed to use q queries, then the error achieved can be bounded by $O(\frac{\log^2 q}{q^2})$, i.e., $\widetilde{O}(\frac{1}{q^2})$.

▶ **Theorem 28.** *There exists an algorithm for* Approximate Convex Hull *in* $\mathbb{R}^2$ *that performs* q *adaptive halfplane emptiness queries and has error* $\widetilde{O}(1/q^2)$.

We remark that replacing the threshold $\frac{1}{q^2}$ in Algorithm 4 with an even smaller number cannot result in an improvement for Theorem 28: while it can decrease the error of the output, it also increases the number of queries substantially. In fact, the bound $\widetilde{O}(1/q^2)$ is already tight up to logarithmic factors. The details are included in the full version.

▶ **Theorem 29.** *Any deterministic algorithm for* Approximate Convex Hull *in* $\mathbb{R}^2$ *that performs* q *adaptive halfplane emptiness queries has error* $\Omega(1/q^2)$.

References

1. Peyman Afshani and Timothy M Chan. On approximate range counting and depth. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 337–343, 2007. doi:10.1145/1247069.1247129.
2. Peyman Afshani and Timothy M Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 180–186. SIAM, 2009. doi:10.1137/1.9781611973068.21.
3. Pankaj K Agarwal. Range searching. In *Handbook of discrete and computational geometry*, pages 1057–1092. Chapman and Hall/CRC, 2017.
4. Pankaj K Agarwal, Jeff Erickson, et al. Geometric range searching and its relatives. *Contemporary Mathematics*, 223(1):56, 1999.
5. Pankaj K. Agarwal and Jirí Matousek. On range searching with semialgebraic sets. *Discrete & Computational Geometry*, 11(4):393–418, 1994. doi:10.1007/BF02574015.
6. Pankaj K Agarwal and Jiří Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13(4):325–345, 1995. doi:10.1007/BF01293483.
7. David Avis and David Bremner. How good are convex hull algorithms? In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 20–28, 1995. doi:10.1145/220279.220282.
8. C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996. doi:10.1145/235815.235821.
9. Jon Louis Bentley and Hermann A. Maurer. Efficient worst-case data structures for range searching. *Acta Informatica*, 13(2):155–168, 1980. doi:10.1007/BF00263991.
10. Kevin Q Brown. Voronoi diagrams from convex hulls. *Information processing letters*, 9(5):223–228, 1979. doi:10.1016/0020-0190(79)90074-7.
11. Timothy M Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & computational geometry*, 16(4):361–368, 1996. doi:10.1007/BF02712873.
12. Timothy M Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the ram, revisited. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 1–10, 2011.

13 Bernard Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *Journal of the ACM (JACM)*, 37(2):200–212, 1990. doi:10.1145/77600.77614.

14 Bernard Chazelle, Ding Liu, and Avner Magen. Sublinear geometric algorithms. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 531–540, 2003. doi:10.1145/780542.780620.

15 Artur Czumaj, Funda Ergün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Approximating the weight of the euclidean minimum spanning tree in sublinear time. *SIAM Journal on Computing*, 35(1):91–109, 2005. doi:10.1137/S0097539703435297.

16 Artur Czumaj and Christian Sohler. Property testing with geometric queries. In *European Symposium on Algorithms*, pages 266–277. Springer, 2001. doi:10.1007/3-540-44676-1_22.

17 Anne Driemel, Morteza Monemizadeh, Eunjin Oh, Frank Staals, and David P. Woodruff. Range Counting Oracles for Geometric Problems. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:16, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2025.42.

18 Herbert Edelsbrunner and Emo Welzl. Halfplanar range search in linear space and o (n0. 695) query time. *Information processing letters*, 23(5):289–293, 1986. doi:10.1016/0020-0190(86)90088-8.

19 Jeff Erickson. Space-time tradeoffs for emptiness queries. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 304–313, 1997.

20 David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. In *Proceedings of the second annual symposium on Computational geometry*, pages 61–71, 1986. doi:10.1145/10515.10522.

21 Jiří Matoušek. Geometric range searching. *ACM Computing Surveys (CSUR)*, 26(4):422–461, 1994. doi:10.1145/197405.197408.

22 Morteza Monemizadeh. Facility Location in the Sublinear Geometric Model. In Nicole Megow and Adam Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:24, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.6.

23 Edgar A Ramos. On range reporting, ray shooting and k-level construction. In *Proceedings of the fifteenth annual symposium on Computational Geometry*, pages 390–399, 1999. doi:10.1145/304893.304993.

24 Raimund Seidel. *A convex hull algorithm optimal for point sets in even dimensions*. PhD thesis, University of British Columbia, 1981.

25 Raimund Seidel. Convex hull computations. In *Handbook of discrete and computational geometry*, pages 687–703. Chapman and Hall/CRC, 2017.

26 Micha Sharir and Hayim Shaul. Semialgebraic range reporting and emptiness searching with applications. *SIAM Journal on Computing*, 40(4):1045–1074, 2011. doi:10.1137/090765092.

27 Andrew Chi-Chih Yao. A lower bound to finding convex hulls. *Journal of the ACM (JACM)*, 28(4):780–787, 1981. doi:10.1145/322276.322289.

Better Sampling Bounds for Restricted Delaunay Triangulations and a Star-Shaped Property for Restricted Voronoi Cells

Jonathan Richard Shewchuk
University of California, Berkeley, CA, USA

Abstract

The restricted Delaunay triangulation of a closed surface Σ and a finite point set $V \subset \Sigma$ is a subcomplex of the Delaunay tetrahedralization of V whose triangles approximate Σ. It is well known that if V is a sufficiently dense sample of a smooth Σ, then the union of the restricted Delaunay triangles is homeomorphic to Σ. We show that an ϵ-sample with $\epsilon \leq 0.3245$ suffices. By comparison, Dey proves it for a 0.18-sample; our improved sampling bound reduces the number of sample points required by a factor of 3.25. More importantly, we improve a related sampling bound of Cheng et al. for Delaunay surface meshing, reducing the number of sample points required by a factor of 21. The first step of our homeomorphism proof is particularly interesting: we show that for a 0.44-sample, the restricted Voronoi cell of each site $v \in V$ is homeomorphic to a disk, and the orthogonal projection of the cell onto $T_v\Sigma$ (the plane tangent to Σ at v) is star-shaped.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Restricted Delaunay triangulation, restricted Voronoi diagram, surface sampling, surface mesh generation, surface reconstruction, ϵ-sample, homeomorphism

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.90

Related Version *Full Version*: `https://arxiv.org/abs/2603.19826` [32]

Funding Supported by the National Science Foundation under Award CCF-1909204.

Acknowledgements I thank Nina Amenta, Jean-Daniel Boissonnat, Siu-Wing Cheng, Tamal Dey, Arijit Ghosh, and Marc Khoury for discussions about surface sampling; and INRIA Sophia-Antipolis and the Geometrica Group, where this work began, for their kind reception during my 2010 sabbatical.

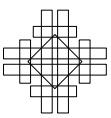
1 Introduction

The *restricted Delaunay triangulation* (RDT) is a well-established way of generating good-quality triangulations on curved surfaces [23]. Researchers have developed a theory of surface sampling to determine how we should sample points on a surface to guarantee that an RDT (or a related triangulation) is a topologically correct and geometrically accurate approximation of the surface [1, 2, 3, 6, 10, 12, 13, 14, 15, 17, 16, 18, 20, 23, 24, 26, 31, 34, 35]. RDTs and this surface sampling theory have equipped geometers to rigorously prove the correctness of algorithms for surface reconstruction [20] and surface mesh generation [18].

Think of the RDT as a function that takes in two inputs: a smooth, closed (compact with no boundary) surface $\Sigma \subset \mathbb{R}^3$ and a finite set $V \subset \Sigma$ of points, called *sites* (or *vertices* of the RDT). The set V is a *sample* or a *point cloud*. The output is a simplicial complex $\mathcal{T}$ whose vertices are V. The RDT $\mathcal{T}$ is a subcomplex of the three-dimensional Delaunay triangulation $\operatorname{Del} V$, but in typical usage $\mathcal{T}$ contains no tetrahedra; only triangles, edges, and the vertices V.

If V is sufficiently dense, $\mathcal{T}$ is a (topological) *triangulation* of Σ, which means that the *underlying space* of $\mathcal{T}$, written $|\mathcal{T}| = \bigcup_{\tau \in \mathcal{T}} \tau$, is homeomorphic to Σ. This paper proves that a modest sampling requirement suffices to guarantee that homeomorphism.


© Jonathan Richard Shewchuk;
licensed under Creative Commons License CC-BY 4.0



What does it mean for V to be "sufficiently dense"? Intuitively, there should be no large unsampled bare spots on Σ. Ideally, sampling requirements are adaptive, as a denser spacing of sites is needed in regions where Σ has higher curvature or closely-spaced "parts" like fingers on a hand, but simple regions need few sites. Many provably good algorithms for surface reconstruction assume that V is a so-called ϵ-sample of Σ (defined in Section 2), an adaptive sample in which a smaller ϵ implies more sites, closer together.

One main result of this paper is that if V is a 0.3245-sample of Σ, the underlying space of the RDT is homeomorphic to Σ. Dey [20] proved the same for a 0.18-sample. The new result reduces the number of sample points required by a factor of 3.25 (the square of 0.3245/0.18). Some well-known surface reconstruction algorithms such as the Crust [1] and Cocone [3] algorithms rely on identifying a superset of the RDT's triangles then paring them down. Any substantial relaxation of their sampling requirements is good news for a broad swath of existing algorithms, and it helps to explain why they work well in practice.

As a point of comparison, Bjerkevik [5] shows that no proof will ever guarantee homeomorphism for 0.72-samples, as there exist 0.72-samples with multiple, topologically different, correct reconstructions. (This limitation holds also for smooth, closed curves in the plane.)

We define another adaptive sampling condition better suited to mesh generation, enabling stronger sampling bounds: we prove homeomorphism for what we call a 0.4132-*Voronoi sample*. Cheng et al. [18] proved the same for a 0.09-Voronoi sample; our result reduces the number of sample points required by a factor of 21. Our bound implies better sampling bounds for existing surface meshing algorithms; it is this paper's most important result.

Also of interest are the new techniques introduced here to obtain these bounds. In particular, for a 0.44-sample or a 0.78-Voronoi sample, the restricted Voronoi cell of each site $v \in V$ (defined in Section 2) is homeomorphic to a disk. Moreover, let $T_v\Sigma$ denote the plane tangent to Σ at v; the orthogonal projection of v's cell onto $T_v\Sigma$ is *star-shaped*: a union of (infinitely many) line segments terminating at v. It seems a bit surprising that restricted Voronoi cells are better behaved with respect to coarse samples than restricted Voronoi vertices or edges, because the proofs by Dey [20] and Cheng et al. [18] use the soundness of the restricted Voronoi edges to establish the soundness of the restricted Voronoi cells. This paper reverses that sequence and, in my opinion, gets at the heart of the reasons why a restricted Voronoi cell is nicely shaped. Remarkably, the results about the Voronoi cells generalize to manifolds of higher dimension embedded in higher-dimensional spaces with no degradation in the bounds [32], although the homeomorphism result does not and cannot generalize to higher dimensions [16, 11, 9].

Not everyone gets excited by improved constants, but I advocate for the importance of tighter sampling theory for computational geometry – just as numerical analysts have devoted much effort to improving constants associated with quadrature rules, interpolation theory, and more. Strong bounds show that RDTs are useful, not merely theoretical.

2 Restricted Delaunay triangulations and ϵ-samples

The RDT is defined by dualizing a *restricted Voronoi diagram*, which will be our main object of study. Let $|pq|$ denote the Euclidean distance from p to q; equivalently, the length of the line segment pq. Given a closed surface $\Sigma \subset \mathbb{R}^3$ and a sample $V \subset \Sigma$, the *restricted Voronoi cell* of a site $v \in V$ is $\mathrm{Vor}|_\Sigma v = \{p \in \Sigma : |pv| \leq |pw| \text{ for all } w \in V\}$. Equivalently, $\mathrm{Vor}|_\Sigma v = \Sigma \cap \mathrm{Vor}\, v$, where $\mathrm{Vor}\, v$ is v's standard Voronoi cell in $\mathbb{R}^3$. The name "restricted Voronoi cell" means that $\mathrm{Vor}|_\Sigma v$ is the restriction of $\mathrm{Vor}\, v$ to the surface Σ. See Figure 1.

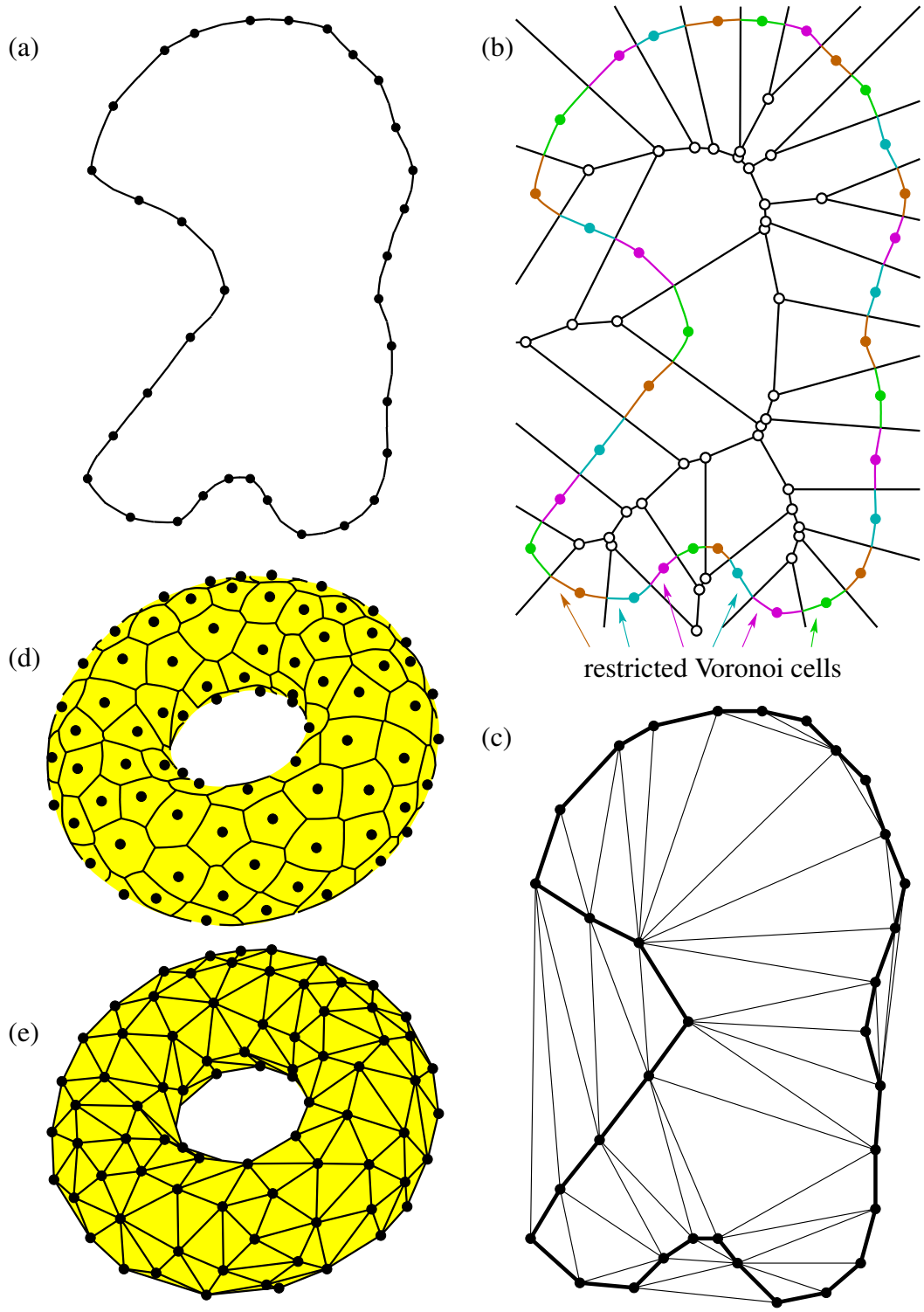


Figure 1 (a) A two-dimensional view of restricted Delaunay triangulations. The input is a smooth, closed curve Σ and a sample $V \subset \Sigma$. (b) The restricted Voronoi diagram is the restriction of the (classic) Voronoi diagram to Σ. (c) The restricted Delaunay triangulation (bold) is the dual of the restricted Voronoi diagram and a subcomplex of the (classic) Delaunay triangulation. (d) A restricted Voronoi diagram in three dimensions. (e) Its dual restricted Delaunay triangulation.

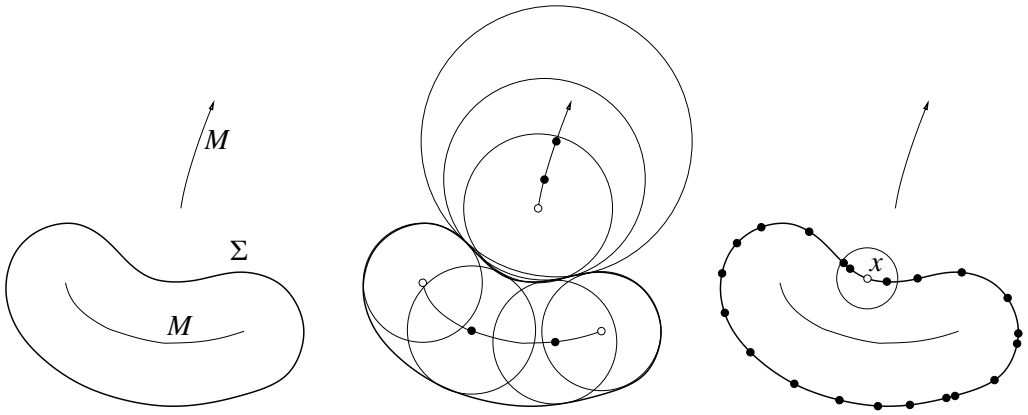


Figure 2 Left: A curve Σ and its medial axis M. Center: Some of the medial balls that define M. Balls with black centers touch two points on Σ. The white points are in the closure of the black centers. Right: A 0.5-sample of Σ (black points). The ball with center x and radius $0.5\,\mathrm{lfs}(x)$ contains a site.

A *restricted Voronoi face* is any nonempty intersection of one or more restricted Voronoi cells – i.e., $\Sigma \cap F$ for some face F in $\mathrm{Vor}\,V$. In particular, a *restricted Voronoi vertex* is the nonempty intersection of Σ with an edge in $\mathrm{Vor}\,V$; and a *restricted Voronoi edge* is the nonempty intersection of Σ with a polygonal 2-face in $\mathrm{Vor}\,V$. Typically these entities are a single point and a (curvy) path on Σ, respectively, but if V is not dense enough, they may take on more pathological forms – e.g., a restricted Voronoi "edge" could be a cycle, a pair of disjoint paths, or even a 2-dimensional blob. This paper aims to determine sampling conditions that eliminate such pathologies. The *restricted Voronoi diagram* $\mathrm{Vor}|_\Sigma V$ is the cell complex containing all the restricted Voronoi faces (including the restricted Voronoi cells).

The *Delaunay subdivision* $\mathrm{Del}\,V$ is the polyhedral complex dual to the Voronoi diagram, and the *restricted Delaunay subdivision* $\mathrm{Del}|_\Sigma V$ is the subcomplex of $\mathrm{Del}\,V$ dual to the restricted Voronoi diagram. That is, for each Voronoi face $F \in \mathrm{Vor}\,V$, let $W \subseteq V$ be the set of sites whose Voronoi cells include F and let F^* be the convex hull of W; we say that F^* is the face *dual* to F. Then $\mathrm{Del}\,V = \{F^* : F \in \mathrm{Vor}\,V\}$. The restricted Delaunay subdivision contains the dual faces whose primal faces intersect Σ; that is, $\mathrm{Del}|_\Sigma V = \{F^* \in \mathrm{Del}\,V : F \cap \Sigma \neq \emptyset\}$. The restricted Voronoi face $f = F \cap \Sigma$ has the same dual face as F, $f^* = F^*$. It is customary to subdivide the polyhedra in $\mathrm{Del}\,V$ into tetrahedra (in which case the duality is no longer strict); accordingly, we can subdivide the polygons in $\mathrm{Del}|_\Sigma V$ into triangles and call it a *restricted Delaunay triangulation*. This paper's results apply whether we do or don't.

A crucial observation in the theory of surface sampling is that the sampling density necessary for accurate approximation is proportional to a field called the *local feature size*. The *medial axis* M of Σ, illustrated in Figure 2, is the closure of the set of all points in $\mathbb{R}^3$ for which the closest point on Σ is not unique. A *medial ball* is a ball whose center lies on M and whose boundary intersects Σ (tangentially), but the interior of the ball does not. For any point $x \in \Sigma$, there are one or two medial balls that have x on their boundaries, called *medial balls at* x. One is inside Σ. If there are two, the other is outside. If not, there is an open halfspace tangent to Σ at x, disjoint from Σ, that we call a degenerate "medial ball."

The *local feature size* function is $\mathrm{lfs} : \Sigma \to \mathbb{R}$, $x \mapsto d(x, M)$ where $d(x, M) = \min_{m \in M} |xm|$. We require that Σ is smooth in the sense that $\inf_{x \in \Sigma} \mathrm{lfs}(x) > 0$. ($C^{1,1}$-continuity suffices.)

A finite point set $V \subset \Sigma$ is an *ϵ-sample of* Σ if for every point $x \in \Sigma$, there is a site $v \in V$ such that $|xv| \leq \epsilon\,\mathrm{lfs}(x)$. That is, the ball with center x and radius $\epsilon\,\mathrm{lfs}(x)$ contains at least one site. A finite $V \subset \Sigma$ is an *ϵ-Voronoi sample of* Σ if $V \neq \emptyset$ and for every site

$v \in V$ and every point $x \in \mathrm{Vor}|_\Sigma v$, $|xv| \leq \epsilon\,\mathrm{lfs}(v)$. That is, $\mathrm{Vor}|_\Sigma v$ is a subset of the ball with center v and radius $\epsilon\,\mathrm{lfs}(v)$. (Note that this condition implies the *ϵ-small condition* of Cheng et al. [18], though the converse is not true.) Only ϵ-samples are a well-known concept, but ϵ-Voronoi samples are nice because $\mathrm{lfs}(v)$ tends to give better sampling bounds than $\mathrm{lfs}(x)$.

Like the classical proofs [1, 18, 20], this paper's homeomorphism proofs rely on the Topological Ball Theorem of Edelsbrunner and Shah [23]. Given a sample $V \subset \Sigma$ of a closed surface $\Sigma \subset \mathbb{R}^3$, this theorem states that $|\mathrm{Del}|_\Sigma V|$ is homeomorphic to Σ if Σ and V satisfy two properties. The *closed ball property* is that for each Voronoi face $F \in \mathrm{Vor}\,V$, $f = F \cap \Sigma$ is either empty or a topological closed $(k-1)$-ball where k is the dimension of F. That is,

A. for a Voronoi 3-cell F, f is a topological closed disk;
B. for a Voronoi 2-face F, f is a topological closed interval or $\emptyset$;
C. for a Voronoi edge F, f contains at most one point; and
D. for a Voronoi vertex F, $f = \emptyset$.

If A–D hold, the *generic intersection property* is that for each $F \in \mathrm{Vor}\,V$, $\mathrm{int}\ f \subset \mathrm{int}\ F$ and $\mathrm{bd}\ f \subset \mathrm{bd}\ F$. In this definition, "interior" and "boundary" are interpreted by the rules of $(k-1)$-manifolds (for f) and k-manifolds (for F). (They are *not* the interior and boundary with respect to $\mathbb{R}^3$.) For a smooth Σ, the generic intersection property holds if properties A–D hold and there is no face $F \in \mathrm{Vor}\,V$ and no point $x \in F \cap \Sigma$ such that $F \subset T_x\Sigma$. That is,

E. there is no point where Σ intersects a 2-face of $\mathrm{Vor}\,V$ tangentially; and
F. there is no point where Σ intersects an edge of $\mathrm{Vor}\,V$ tangentially.

This paper is organized around proving that these conditions hold for a sufficiently dense sample V: properties A and E in Section 4, properties C and F in Section 5, and property B in Section 6. Property D ensures that $\mathrm{Del}|_\Sigma V$ contains no polyhedra – only faces of dimension two or less. Property D cannot be enforced by dense sampling, but it can be enforced by an infinitesimal perturbation of Σ so that Σ intersects no Voronoi vertex. This perturbation is easy to simulate symbolically in software simply by treating each Voronoi vertex on Σ as if it were strictly inside Σ (when deciding which faces of $\mathrm{Del}\,V$ are in $\mathrm{Del}|_\Sigma V$).

Unlike in Dey [20] or Cheng et al. [18], our proof of property A does not rely on property B or C. Property A holds for a 0.4401-sample or a 0.7861-Voronoi sample, whereas we prove property B only for a 0.3245-sample or a 0.4132-Voronoi sample. Property B (restricted Voronoi edges) is the bottleneck that determines our sampling requirements.

Some algorithms for triangulating surfaces guarantee topological correctness without RDTs, by other methods [4, 8, 19, 27, 29]. One alternative is to consider Voronoi diagrams with other distance metrics. Dyer, Zhang, and Möller [21, 22] and others [28, 7] use intrinsic distances within Σ, also known as geodesic distances when Σ is smooth, to define *intrinsic Voronoi diagrams* that dualize to *intrinsic Delaunay triangulations* (IDTs). Advantages are that the Voronoi cells are trivially star-shaped, and the sampling requirements needed to guarantee a homeomorphic triangulation are mild [22]. But intrinsic distances on smooth surfaces are painful to compute [30]. Restricted Delaunay triangulations will probably remain popular as an easy alternative. So let us see how far we can push them.

3 A relationship between surface points and nearby tangent planes

For a smooth, closed surface Σ and a point $x \in \Sigma$, let $T_x\Sigma \subset \mathbb{R}^3$ denote the plane tangent to Σ at x. ($T_x\Sigma$ passes through x; not necessarily through the origin.) Lemma 1, below, establishes a relationship between $T_x\Sigma$ and v for two nearby points $v, x \in \Sigma$. This relationship prepares us to prove in Section 4 that under suitable sampling conditions, a restricted Voronoi cell is a topological disk with a star-shaped projection on its site's tangent plane. Lemma 1 is surprisingly strong; the constant ξ (as it is applied in Theorem 6) will likely be hard to beat.

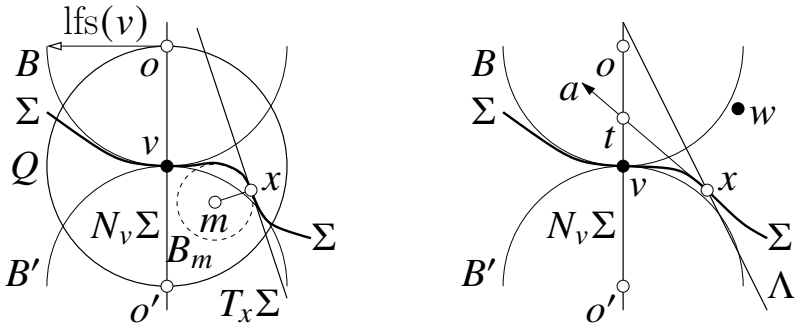


Figure 3 Left: for two nearby points $v, x \in \Sigma$, suppose that the tangent plane $T_x\Sigma$ does not intersect oo', as shown. This leads to a contradiction; hence $T_x\Sigma$ must intersect oo'. The medial ball B_m is tangent to Σ at x. The center m of B_m cannot lie in the open ball Q. The points v, x, o, and o' lie on the plane of the page, but m and w generally do not; imagine m floating above the page. The dashed circle shows the page's cross section of B_m, but B_m is larger. The surface Σ cannot intersect the open balls B, B', and B_m, so B and B_m are disjoint. Right: the plane Λ bisects vw. The ray $a \in T_x\Sigma$ intersects the relative interior of oo' and is strictly on the same side of Λ as v.

Let B and B' be the two open balls of radius $\operatorname{lfs}(v)$ tangent to Σ at v, and let o and o' be their centers. As B and B' are subsets of the open medial balls at v, they are disjoint from Σ.

▶ **Lemma 1.** *Consider two points $v, x \in \Sigma$ such that $|vx| < \xi \operatorname{lfs}(v)$, where $\xi = \sqrt{(\sqrt{5}-1)/2} \doteq 0.786151$. Then o and o' lie on strictly opposite sides of $T_x\Sigma$.*

Proof. Suppose for the sake of contradiction that o and o' do not lie on strictly opposite sides of $T_x\Sigma$, as illustrated in Figure 3, left. Then $T_x\Sigma \neq T_v\Sigma$ and $x \neq v$. Moreover, either $oo' \subset T_x\Sigma$ or $T_x\Sigma$ does not intersect the relative interior of oo'.

Let B_m be the open medial ball tangent to Σ at x that is on the same side of $T_x\Sigma$ as v (either side if $v \in T_x\Sigma$), as illustrated. If B_m is an open halfspace then $T_x\Sigma = \text{boundary } B_m$, $v \in T_x\Sigma$ (as $v \notin B_m$), $T_v\Sigma = T_x\Sigma$ (as $\Sigma \cap B_m = \emptyset$), and the lemma follows. So assume B_m is bounded. Its center m lies on Σ's medial axis. The line segment xm is perpendicular to $T_x\Sigma$.

Observe that v is in the relative interior of oo'. If $oo' \subset T_x\Sigma$, then $\angle vxm = \angle oxm = \angle o'xm = 90°$. Otherwise, $T_x\Sigma$ does not intersect the relative interior of oo', so $\angle vxm < 90°$, $\angle oxm \leq 90°$, and $\angle o'xm \leq 90°$ (the last two because o and o' cannot be on the side of $T_x\Sigma$ opposite from v). By Pythagoras' Theorem, $|ox|^2 + |mx|^2 \geq |om|^2$ and $|vx|^2 + |mx|^2 \geq |vm|^2$. As m lies on the medial axis, $|vm| \geq \operatorname{lfs}(v)$ (by the definition of lfs), so $|vx|^2 + |mx|^2 \geq \operatorname{lfs}(v)^2$.

The surface Σ intersects none of the open balls B, B', or B_m, but it passes between B and B' at v. As Σ is closed and cuts space into two pieces, one containing B and one containing B', the ball B_m must lie in one of those two pieces. Choose the labels B and B' so that B_m lies in the same piece as B', as illustrated; then B_m must be disjoint from B. The radii of B and B_m are $\operatorname{lfs}(v)$ and $|mx|$ respectively, so $|om| \geq \operatorname{lfs}(v) + |mx|$. Combining this with the inequality $|ox|^2 + |mx|^2 \geq |om|^2$ gives $|ox|^2 \geq \operatorname{lfs}(v)^2 + 2\operatorname{lfs}(v)\,|mx|$. Combining this with the inequality $|mx|^2 \geq \operatorname{lfs}(v)^2 - |vx|^2$ gives $|ox|^2 \geq \operatorname{lfs}(v)^2 + 2\operatorname{lfs}(v)\sqrt{\operatorname{lfs}(v)^2 - |vx|^2}$.

Let $N_v\Sigma$ be the line through o', v, and o (the vertical axis in Figure 3). Create a coordinate system with $v = (0,0,0)$ and $x = (x_h, x_v, 0)$ such that x_v is the coordinate in the direction parallel to $N_v\Sigma$ and x_h is the distance from $N_v\Sigma$ to x (the horizontal axis in Figure 3). Then $|ox|^2 + |o'x|^2 = x_h^2 + (x_v - \operatorname{lfs}(v))^2 + x_h^2 + (x_v + \operatorname{lfs}(v))^2 = 2x_h^2 + 2x_v^2 + 2\operatorname{lfs}(v)^2 = 2\,|vx|^2 + 2\operatorname{lfs}(v)^2$. Rewrite this as $|vx|^2 = (|ox|^2 + |o'x|^2 - 2\operatorname{lfs}(v)^2)/2$. As $x \notin B'$, $|o'x|^2 \geq \operatorname{lfs}(v)^2$. Combining these with the inequality $|ox|^2 \geq \operatorname{lfs}(v)^2 + 2\operatorname{lfs}(v)\sqrt{\operatorname{lfs}(v)^2 - |vx|^2}$ gives $|vx|^2 \geq \operatorname{lfs}(v)\sqrt{\operatorname{lfs}(v)^2 - |vx|^2}$.

As $|vx| < \xi \operatorname{lfs}(v)$, we have $\xi^2 > |vx|^2/\operatorname{lfs}(v)^2 \geq \sqrt{1 - |vx|^2/\operatorname{lfs}(v)^2} > \sqrt{1-\xi^2}$, which is equivalent to $\xi^4 + \xi^2 - 1 > 0$, hence $\xi > \sqrt{(\sqrt{5}-1)/2}$. The result follows by contradiction. ◀

4 Restricted Voronoi cells are topological disks

This section investigates sampling conditions that guarantee that (1) every restricted Voronoi cell has the topology of a closed disk (closed ball property A), (2) the projection of each restricted Voronoi cell onto its site's tangent plane is star-shaped, and (3) no 2-face in $\operatorname{Vor} V$ intersects Σ tangentially (generic intersection property E). Theorem 6 shows that a 0.78-Voronoi sample suffices, and Corollary 9 shows that a 0.44-sample suffices.

Consider a site $v \in V$, its Voronoi cell $\operatorname{Vor} v$, and its restricted Voronoi cell $\operatorname{Vor}|_\Sigma v = \Sigma \cap \operatorname{Vor} v$. Let φ be the map that orthogonally projects $\mathbb{R}^3$ onto $T_v\Sigma$. Note that $\varphi(v) = v$.

Define a *radial path* to be a topological closed interval $\gamma \subset \operatorname{Vor}|_\Sigma v$ such that

1. one endpoint of γ is the site v,
2. the other endpoint – call it z – lies on the boundary of $\operatorname{Vor} v$,
3. every point on $\gamma \setminus \{z\}$ lies in the interior of $\operatorname{Vor} v$, and
4. $\varphi|_\gamma$ is a homeomorphism from γ to a line segment on $T_v\Sigma$ with endpoints v and $\varphi(z)$, where $\varphi|_\gamma$ denotes the restriction of φ to the domain γ.

We will see that under suitable sampling conditions, every point in $\operatorname{Vor}|_\Sigma v$ lies on exactly one radial path, except v itself (Lemma 4). It follows that we can decompose $\operatorname{Vor}|_\Sigma v$ into radial paths such that no two share a point besides v (Lemma 5). That is, if we remove v from each radial path, then we have a partition of $\operatorname{Vor}|_\Sigma v \setminus \{v\}$ into paths. Therefore, $\varphi(\operatorname{Vor}|_\Sigma v)$ is star-shaped. Although $\operatorname{Vor}|_\Sigma v$ itself is not star-shaped, its decomposition into radial paths is a curvy variant of "star-shaped." As the lengths of the projected radial paths vary continuously with their polar angles, $\operatorname{Vor}|_\Sigma v$ is homeomorphic to a closed disk (Theorem 6).

Let $N_v\Sigma$ be the line normal to Σ at v (orthogonal to $T_v\Sigma$). Let B and B' be the two open balls of radius $\operatorname{lfs}(v)$ tangent to Σ at v, and let o and o' be their centers. Then $o, o' \in N_v\Sigma$.

The following lemma implies that if you are standing on the boundary of $\operatorname{Vor}|_\Sigma v$ and you walk toward v on a radial path, you immediately enter the interior of $\operatorname{Vor} v$. (The proof of Lemma 4 develops this idea further.) It also implies generic intersection property E.

▶ **Lemma 2.** *Consider two distinct sites $v, w \in V$ and a point $x \in \operatorname{Vor}|_\Sigma v \cap \operatorname{Vor}|_\Sigma w$. Suppose that $|vx| < \xi \operatorname{lfs}(v)$ where $\xi = \sqrt{(\sqrt{5}-1)/2} \doteq 0.786151$. Let Λ be the plane that orthogonally bisects the line segment vw (thus $x \in \Lambda$). By Lemma 1, $T_x\Sigma$ intersects the relative interior of the line segment oo' at a lone point t. Let a be the open ray $\vec{xt}$, and observe that $a \subset T_x\Sigma$.*

Then v and a are strictly on the same side of Λ.

Proof. See Figure 3, right. Neither B nor B' intersects Σ, hence neither ball contains w, hence $|vo| \le |wo|$ and $|vo'| \le |wo'|$. Therefore, each of o and o' lies either on Λ or on the same side of Λ as v, as illustrated. (More broadly, $oo' \subset \operatorname{Vor} v$.) As $v \in oo'$ and $v \notin \Lambda$, Λ does not intersect the relative interior of oo'. By contrast, a does intersect the relative interior of oo' (at t). Recall that a's origin x lies on Λ. Therefore, the open ray a is strictly on the same side of Λ as the relative interior of oo', which contains v. ◀

Lemma 4, below, shows that under suitable sampling conditions, every point in $\operatorname{Vor}|_\Sigma v \setminus \{v\}$ lies on one and only one radial path. It depends on the simple observation of Lemma 3.

▶ **Lemma 3.** *Let $v, x \in \Sigma$ be two points such that $|vx| < \xi \operatorname{lfs}(v)$. There exists an open neighborhood $N \subset \Sigma$ of x such that $\varphi|_N$ is a homeomorphism from N to its image $\varphi(N) \subset T_v\Sigma$.*

Proof. As $|vx| < \xi \operatorname{lfs}(v)$, by Lemma 1 (or Lemma 13), $T_x\Sigma$ is not perpendicular to $T_v\Sigma$. It follows from the smoothness of Σ that if N is sufficiently small, $\varphi|_N$ is injective. As $\varphi|_N$ is injective and both $\varphi|_N$ and its inverse are continuous, $\varphi|_N$ is a homeomorphism. ◀

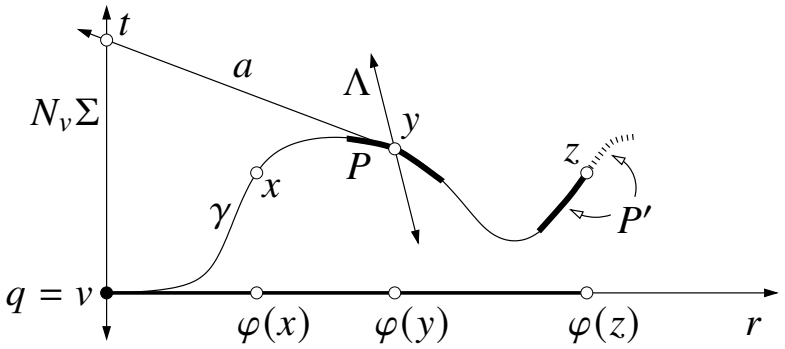


Figure 4 A radial path γ with endpoints v and z. (The path P' extends past γ on Σ.)

▶ **Lemma 4.** *Consider a site $v \in V$, its restricted Voronoi cell $C = \mathrm{Vor}|_\Sigma v$, and a point $x \in C \setminus \{v\}$. Let $r \in T_v\Sigma$ be the closed ray with origin v that passes through $\varphi(x)$. Suppose that for every point $y \in C$, $|vy| < \xi\,\mathrm{lfs}(v)$, where $\xi = \sqrt{(\sqrt{5}-1)/2} \doteq 0.786151$.*

Then there is a unique radial path $\gamma \subset C$ such that $x \in \gamma$. Furthermore, γ is the only radial path such that $\varphi(\gamma) \subseteq r$.

Proof. For every point $y \in C \setminus \{v\}$ (including x), $\varphi(y) \neq v$, as if $\varphi(y) = v$ we have a contradiction: $|vy| < \xi\,\mathrm{lfs}(v)$ and $y \neq v$ imply that y is in B or B' and thus not in C.

Define the point set $\varphi|_C^{-1}(r) = \{y \in C : \varphi(y) \in r\}$ (the intersection of C with the closed halfplane with boundary $N_v\Sigma$, passing through x). Clearly, $x \in \varphi|_C^{-1}(r)$. Let γ be the connected component of $\varphi|_C^{-1}(r)$ that contains x. We will show that γ is a radial path.

As $\gamma \subset C$, for every point $y \in \gamma$, $|vy| < \xi\,\mathrm{lfs}(v)$ and by Lemma 3 there exists an open neighborhood $N \subset \Sigma$ of y such that $\varphi|_N$ is a homeomorphism, so $\varphi|_{N\cap\gamma}$ is a homeomorphism from $N \cap \gamma$ to its image $\varphi(N \cap \gamma)$. In other words, $\varphi|_\gamma$ is a local homeomorphism from γ to its image $\varphi(\gamma)$. As γ is connected and $\varphi(\gamma)$ is embedded in the ray r, $\varphi|_\gamma$ is a (global) homeomorphism. (Intuitively, the map φ cannot cause the path to double back on itself, so $\varphi|_\gamma$ is an injection.) Therefore, γ is a topological interval or a lone point.

As C is compact and r is closed, $\varphi|_C^{-1}(r)$ is compact and γ is compact. Let q and z be the endpoints of γ, chosen so that $|v\varphi(q)| \leq |v\varphi(z)|$; see Figure 4. As $\varphi|_\gamma$ is a homeomorphism, $\varphi(q)$ and $\varphi(z)$ are the endpoints of $\varphi(\gamma)$. As $\varphi(\gamma)$ contains $\varphi(x)$ and $\varphi(x) \neq v$, $\varphi(z) \neq v$ and $z \neq v$. We will show that $q = v$ (which implies that $\varphi(C)$ is star-shaped) and that z is on the boundary of $\mathrm{Vor}\,v$, thereby establishing the first two criteria for γ to be a radial path.

But first, we show that $\gamma \setminus \{z\}$ is in the interior of $\mathrm{Vor}\,v$, the third criterion for γ to be a radial path. Suppose for the sake of contradiction that a point $y \in \gamma \setminus \{z\}$ lies on the boundary of $\mathrm{Vor}\,v$. Then y also lies in the restricted Voronoi cell $\mathrm{Vor}|_\Sigma w$ of another site $w \in V \setminus \{v\}$, and $y \in \Lambda$ where Λ is the plane that orthogonally bisects vw. Clearly, $y \neq v$. By Lemma 1, $T_y\Sigma$ intersects $N_v\Sigma$ at a lone point t. Let a be the open ray $\vec{yt}$. Observe that $a \subset T_y\Sigma$; moreover, a is tangent to γ at y, as illustrated in Figure 4.

By Lemma 2, v and a are strictly on the same side of Λ. Hence if you walk along γ from y to z – opposite to the direction of a – you enter w's side of Λ at the instant you leave y, as illustrated. This contradicts the fact that $\gamma \in \mathrm{Vor}|_\Sigma v$. So $\gamma \setminus \{z\}$ is in the interior of $\mathrm{Vor}\,v$.

Let us return to the first two criteria for γ to be a radial path. Consider a point $y \in \gamma \setminus \{v\}$. By Lemma 3, there exists an open neighborhood $N \subset \Sigma$ of y such that $\varphi|_N$ is a homeomorphism. Define $\varphi|_N^{-1}(r) = \{p \in N : \varphi(p) \in r\}$ and let P be the connected component of $\varphi|_N^{-1}(r)$ that contains y, illustrated in Figure 4. As $\varphi(y)$ is in the interiors of r and $\varphi(N)$, $\varphi(y)$ is in the interior of $r \cap \varphi(N)$. As $\varphi|_N$ is a homeomorphism, y is in the relative interior of $\varphi|_N^{-1}(r)$, so P is a path (topological interval) on Σ with y in its relative interior.

First, consider the case where y is in the interior of $\operatorname{Vor} v$ (but $y \neq v$). Then we can shrink the open neighborhood N of y so that $N \subset \operatorname{Vor} v$ and thus $N \subset C$, and thereby have $\varphi|_N^{-1}(r) \subseteq \varphi|_C^{-1}(r)$. As γ is the connected component of $\varphi|_C^{-1}(r)$ that contains y, $P \subseteq \gamma$. Hence y is not an endpoint of γ. We have seen that $\gamma \setminus \{z\}$ is in the interior of $\operatorname{Vor} v$; it follows that only v and z can be endpoints of γ. That is, the endpoint q is either v or z. Moreover, as $z \neq v$ is an endpoint of γ, z is on the boundary of $\operatorname{Vor} v$ (establishing the second criterion).

Second, consider the case where $y = z$. In this case, the path P looks like P' in Figure 4. We use this case to show that $q \neq z$. Suppose for the sake of contradiction that $q = z$; then $\gamma = \{z\}$. It follows that as you walk along P from z, you exit the restricted Voronoi cell C in both directions along P. Let a be the open ray $\vec{zt}$ where $\{t\} = T_z\Sigma \cap N_v\Sigma$ (hence $a \subset T_z\Sigma$); a is tangent to P at z. As P exits C, there exists a site $w \in V \setminus \{v\}$ and a plane Λ that orthogonally bisects vw such that $z \in \Lambda$ and a enters w's side of Λ at z. But this contradicts the fact that, by Lemma 2, v and a are strictly on the same side of Λ. Thus $q \neq z$, so $q = v$.

Therefore, the endpoints of γ are v and z, z is on the boundary of $\operatorname{Vor} v$, $\gamma \setminus \{z\}$ is in its interior, and $\varphi|_\gamma$ is a homeomorphism from γ to $v\varphi(z)$. By definition, γ is a radial path.

To see that γ is the *only* radial path containing x, observe that every radial path containing x is a subset of $\varphi|_C^{-1}(r)$, and moreover is a subset of γ (because a radial path is connected). No strict subset of γ can be a radial path, because every connected strict subset of γ is missing either v or a point on the boundary of $\operatorname{Vor} v$.

The reasoning of this proof holds equally well if we replace x with any other point $x' \in \varphi|_C^{-1}(r) \setminus \{v\}$, so every connected component of $\varphi|_C^{-1}(r)$ contains v. Therefore, $\varphi|_C^{-1}(r)$ is connected. Hence $\gamma = \varphi|_C^{-1}(r)$ and no radial path besides γ has its projection on r. ◀

It follows that we can decompose $\operatorname{Vor}|_\Sigma v$ into radial paths. Hence $\varphi(\operatorname{Vor}|_\Sigma v)$ is star-shaped. It also follows that the orthogonal projection φ, restricted to $\operatorname{Vor}|_\Sigma v$, is a homeomorphism.

▶ **Lemma 5.** *Let $v \in V$ be a site and let $C = \operatorname{Vor}|_\Sigma v$ be its restricted Voronoi cell. Suppose that for every point $y \in C$, $|vy| < \xi \operatorname{lfs}(v)$, where $\xi = \sqrt{(\sqrt{5}-1)/2} \doteq 0.786151$.*

Let Γ be the set of all radial paths for all points in $C \setminus \{v\}$.

Then $\bigcup_{\gamma\in\Gamma} \gamma = C$; hence $\varphi(C)$ is star-shaped. Moreover, no two paths in Γ share a common point besides v, and there is a one-to-one correspondence between paths in Γ and points where Σ intersects the boundary of $\operatorname{Vor} v$.

Moreover, $\varphi|_C$ is a homeomorphism from C to its image $\varphi(C)$ on $T_v\Sigma$.

Proof. By Lemma 4, for each point $x \in C \setminus \{v\}$, there is a unique radial path $\gamma \subset C$ such that $x \in \gamma$. Every radial path contains v. Hence $\bigcup_{\gamma\in\Gamma} \gamma = C$. As each $x \in C \setminus \{v\}$ lies on only one radial path, no two paths in Γ share a common point besides v. By definition, each radial path contains exactly one point on the boundary of $\operatorname{Vor} v$. Hence there is a one-to-one correspondence between radial paths and points where Σ intersects the boundary of $\operatorname{Vor} v$.

Let us see that $\varphi|_C$ is an injection. Consider two points $x, y \in C$ such that $\varphi(x) = \varphi(y)$; we will see that $x = y$. For every radial path $\gamma \in \Gamma$, $\varphi|_\gamma$ is a homeomorphism by the definition of radial path, so if x and y lie on the same radial path, then $x = y$. If x and y lie on distinct radial paths, then $x = y = v$, because by Lemma 4, for any two distinct radial paths $\gamma_1, \gamma_2 \in \Gamma$, $\varphi(\gamma_1)$ and $\varphi(\gamma_2)$ lie on two distinct rays with origin v.

As C is compact and $\varphi|_C$ is injective and continuous, $\varphi|_C$ is a homeomorphism [33]. ◀

Lemma 5 shows that $\operatorname{Vor}|_\Sigma v$ is homeomorphic to its image $I_v = \varphi(\operatorname{Vor}|_\Sigma v)$ on $T_v\Sigma$, but what is the shape of I_v? We obtain a homeomorphism from I_v to a closed unit disk on $T_v\Sigma$ by simply scaling each line segment $\varphi(\gamma)$ to have unit length. Thus we arrive at this section's main theorem, Theorem 6, which states that $\operatorname{Vor}|_\Sigma v$ is a topological closed disk. The proof is deferred to the full-length paper [32], but here is a sketch of the ideas.

Let $E = \{\varphi(\gamma) : \gamma \in \Gamma\}$ be the set of the orthogonal projections of the radial paths onto $T_v\Sigma$. Then we can write $I_v = \bigcup_{e\in E} e$, a decomposition of I_v into line segments with endpoint v, no two leaving v in the same direction (but every direction on $T_v\Sigma$ is represented).

For every point $x \in I_v \setminus \{v\}$, let $l(x)$ be the length of the unique line segment in E that contains x. Thus l is a function over the domain $I_v \setminus \{v\}$, but $l(v)$ is not defined. One can show that l is continuous [32]. This is a consequence of two facts: I_v is a compact point set and every point on a line segment $e \in E$ except one endpoint is in the interior of I_v.

Let $\chi : I_v \to T_v\Sigma$ map each line segment in E to a line segment with unit length (while preserving its direction) – specifically, $\chi(x) = v + \frac{1}{l(x)}(x - v)$ for $x \neq v$ and $\chi(v) = v$. Then χ is continuous as l is continuous and positive. As χ is a bijective, continuous map with a continuous inverse, χ is a homeomorphism from I_v to a closed unit disk.

▶ **Theorem 6.** *Let $v \in V$ be a site and let $C = \mathrm{Vor}|_\Sigma v$ be its restricted Voronoi cell. Suppose that for every point $y \in \mathrm{Vor}|_\Sigma v$, $|vy| < \xi \operatorname{lfs}(v)$, where $\xi = \sqrt{(\sqrt{5}-1)/2} \doteq 0.786151$.*

Then $\chi \circ \varphi|_C$ is a homeomorphism from $\mathrm{Vor}|_\Sigma v$ to a closed unit disk on $T_v\Sigma$.

If we impose the condition of Theorem 6 on *all* the restricted Voronoi cells, every connected component of Σ has at least six sites on it. Lemma 7 follows easily from Lemmas 4 and 13, but there isn't quite enough space here for the proof; see the full-length paper [32].

▶ **Lemma 7.** *Let V be an ϵ-Voronoi sample of Σ for some $\epsilon < \xi = \sqrt{(\sqrt{5}-1)/2} \doteq 0.786151$. Every connected component of Σ has at least six sites and six restricted Voronoi cells on it.*

To apply Theorem 6 to ϵ-samples, observe that 0.44-samples are 0.786-Voronoi samples.

▶ **Lemma 8** (Feature Translation Lemma [3, 20])**.** *Let $\Sigma \subset \mathbb{R}^3$ be a smooth, closed surface and let $p, q \in \Sigma$ be points such that $|pq| \leq \epsilon \operatorname{lfs}(p)$ for some $\epsilon < 1$. Then*

$$\operatorname{lfs}(p) \leq \frac{1}{1-\epsilon}\operatorname{lfs}(q) \quad \text{and} \quad |pq| \leq \frac{\epsilon}{1-\epsilon}\operatorname{lfs}(q).$$

▶ **Corollary 9.** *Let V be an ϵ-sample of Σ for $\epsilon < \frac{\xi}{\xi+1} \doteq 0.440137$.*

Then every restricted Voronoi cell in $\mathrm{Vor}|_\Sigma V$ is homeomorphic to a closed disk (closed disk property A), Σ does not intersect any 2-face of $\mathrm{Vor}\, V$ tangentially (generic intersection property E), every connected component of Σ has at least six sites and six restricted Voronoi cells on it, and no restricted Voronoi cell intersects the interior of another.

Proof. Consider any site $v \in V$ and any point $x \in \mathrm{Vor}|_\Sigma v$. As V is an ϵ-sample, $|vx| \leq \epsilon \operatorname{lfs}(x)$. By the Feature Translation Lemma (Lemma 8), $|vx| \leq \frac{\epsilon}{1-\epsilon} \operatorname{lfs}(v) < \xi \operatorname{lfs}(v)$.

The first three claims follow from Theorem 6 and Lemmas 2 and 7, respectively. The final claim follows from Lemma 5 because every point in the interior of $\mathrm{Vor}|_\Sigma v$ is in the interior of $\mathrm{Vor}\, v$ (by the definition of radial path) and cannot be shared with another cell. ◀

5 Restricted Voronoi vertices are lone points

A restricted Voronoi vertex is a nonempty intersection of Σ with an edge in $\mathrm{Vor}\, V$. However, without suitable sampling conditions, such an intersection might contain many points, even infinitely many. This section describes sampling conditions that guarantee closed ball property C: an intersection of three distinct restricted Voronoi cells contains at most one point, thereby justifying the name "vertex." Generic intersection property F comes as a byproduct. Lemma 10 shows that a 0.49-Voronoi sample suffices, and Corollary 11 shows

that a 0.33-sample suffices (which follows from Lemmas 10 and 8). Both proofs are postponed to the full-length paper [32], as restricted Voronoi vertices are not the bottleneck limiting our homeomorphism theorems' sample bounds. (Restricted Voronoi edges are the bottleneck; see Section 6.) Note that Cheng et al. [18] prove Lemma 10 for a 0.15-Voronoi sample.

▶ **Lemma 10.** *Consider three distinct sites $v, v', v'' \in V$ and the triangle $\tau = \triangle vv'v''$, where v is the vertex at τ's largest plane angle. Let $f = \mathrm{Vor}|_\Sigma v \cap \mathrm{Vor}|_\Sigma v' \cap \mathrm{Vor}|_\Sigma v''$, the restricted Voronoi face dual to τ. Let $\ell_\tau \subset \mathbb{R}^3$ be the line containing all points equidistant to the sites v, v', and v'' (thus $f \subset \ell_\tau$). Suppose that for every point $y \in f$, $|vy| < \kappa\,\mathrm{lfs}(v)$, where κ is the positive real root of $\kappa^4 = 4(1-\kappa^2)(1-\sqrt{3}\kappa)^2$, with approximate value $\kappa \doteq 0.495683$.*

Then f contains at most one point (i.e., a restricted Voronoi vertex). Moreover, if f contains a point, Σ is not tangent to ℓ_τ at that point.

▶ **Corollary 11.** *Let V be an ϵ-sample of Σ for $\epsilon < \frac{\kappa}{\kappa+1} \doteq 0.331409$. Consider three distinct sites $v, v', v'' \in V$ and let $f = \mathrm{Vor}|_\Sigma v \cap \mathrm{Vor}|_\Sigma v' \cap \mathrm{Vor}|_\Sigma v''$. Let $\ell_\tau \subset \mathbb{R}^3$ be the line containing all points equidistant to the sites v, v', and v''.*

Then f contains at most one point, and Σ is not tangent to ℓ_τ at that point.

6 Restricted Voronoi edges are topological intervals

Theorem 6 and Corollary 9 give conditions under which restricted Voronoi cells are topological closed disks. Lemma 10 and Corollary 11 give conditions under which the intersection of three distinct restricted Voronoi cells is at most one point. What about an intersection of two distinct restricted Voronoi cells? That could be empty, a restricted Voronoi vertex, or a *restricted Voronoi edge*, the last being a nonempty intersection of Σ with a 2-face of $\mathrm{Vor}\,V$. Under suitable sampling conditions, Lemma 17, below, guarantees closed ball property B: each restricted Voronoi edge is a topological interval.

The next six lemmas derive conditions in which an intersection $\mathrm{Vor}|_\Sigma v \cap \mathrm{Vor}|_\Sigma w$ is one interval or one isolated point. Lemma 15 applies to 0.32-samples, whereas Lemma 16 applies to 0.41-Voronoi samples, and Lemma 17 concludes both. Lemma 12 has no sampling requirements, only topological requirements; see the full-length paper for a proof [32].

▶ **Lemma 12.** *Let $\mathrm{Vor}|_\Sigma V$ be a restricted Voronoi diagram. Suppose that every restricted Voronoi cell is a topological closed disk and every intersection of three distinct restricted Voronoi cells is either empty or a lone point (henceforth called a restricted Voronoi vertex). Suppose also that no restricted Voronoi cell intersects the interior of another.*

Then for every pair of distinct sites $v, w \in V$, $\mathrm{Vor}|_\Sigma v \cap \mathrm{Vor}|_\Sigma w$ is one of these three: empty; a topological circle containing no restricted Voronoi vertex; or a union of disjoint topological closed intervals and isolated points, where each isolated point is a restricted Voronoi vertex and each interval contains exactly two restricted Voronoi vertices which are its endpoints.

Moreover, if each connected component of Σ has at least three sites in V lying on it, then the possibility that $\mathrm{Vor}|_\Sigma v \cap \mathrm{Vor}|_\Sigma w$ is a topological circle is eliminated, every restricted Voronoi cell has at least two restricted Voronoi vertices on its boundary, and every connected component of Σ has at least two restricted Voronoi vertices on it.

A closed Σ cuts space into two pieces, a bounded *inside* piece and an unbounded *outside* piece. For a point $x \in \Sigma$, let n_x denote the outside-facing vector normal to Σ at x (and normal to $T_x\Sigma$). Let $\angle(n_p, n_q) \in [0°, 180°]$ denote the angle separating two vectors.

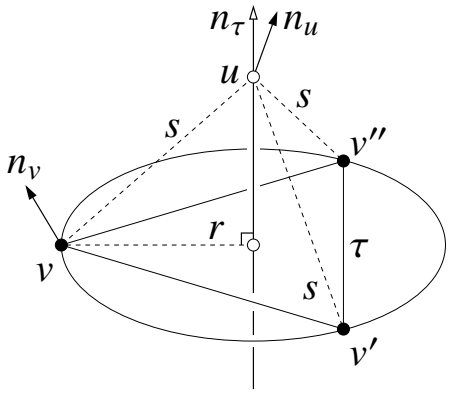


Figure 5 The sites v, v', and v'' and the restricted Voronoi vertex u lie on Σ (not shown).

▶ **Lemma 13** (Normal Variation Lemma [25]). *Consider two points $p, q \in \Sigma$ and let $\delta = |pq|/\mathrm{lfs}(p)$. If $\delta < \sqrt{4\sqrt{5}-8} \doteq 0.971736$, then $\angle(n_p, n_q) \leq \eta(\delta)$ where*

$$\eta(\delta) = \arccos\left(1 - \frac{\delta^2}{2\sqrt{1-\delta^2}}\right) \approx \delta + \frac{7}{24}\delta^3 + O(\delta^5) \text{ radians.}$$

▶ **Lemma 14** (Triangle Normal Lemma [25]). *Let τ be a triangle whose vertices lie on Σ. Let r be the radius of τ's circumscribing circle. Let v be a vertex of τ and let ϕ be τ's plane angle at v. Let n_τ be a vector normal to τ. Let $\mathrm{aff}\,\tau$ denote the affine hull of τ. Then*

$$\sin\angle(n_\tau, n_v) = \sin\angle(\mathrm{aff}\,\tau, T_v\Sigma) \leq \frac{r}{\mathrm{lfs}(v)} \max\left\{\cot\frac{\phi}{2}, 1\right\}.$$

▶ **Lemma 15.** *Let u be a restricted Voronoi vertex and let $\tau = \triangle vv'v''$ be its dual restricted Delaunay triangle. (If u's dual face is a polygon, you may choose any three of its vertices.) Let n_τ be a vector normal to τ, directed so that $\angle(n_v, n_\tau) \leq 90°$ (as illustrated in Figure 5). Let $s = |vu| = |v'u| = |v''u|$ and suppose that $s \leq 0.3245\,\mathrm{lfs}(u)$.*

Then $\angle(n_u, n_\tau) < 90°$ and $\angle(n_v, n_\tau) < 90°$. Equivalently, $n_u \cdot n_\tau$ and $n_v \cdot n_\tau$ are positive.

Proof. Let $w \in \{v, v', v''\}$ be the vertex at τ's largest plane angle. As $|vu| = |wu| = s \leq 0.3245\,\mathrm{lfs}(u)$, by the Normal Variation Lemma (Lemma 13), $\angle(n_v, n_u) \leq \eta(0.3245) < 19.21°$ and $\angle(n_w, n_u) \leq \eta(0.3245) < 19.21°$.

Let r be τ's circumradius. As Figure 5 illustrates, $r \leq s$, because u lies on the line perpendicular to τ through τ's circumcenter and r is the distance from v to that line. By the Feature Translation Lemma (Lemma 8), $\mathrm{lfs}(u) \leq \mathrm{lfs}(v)/(1 - 0.3245)$ and $\mathrm{lfs}(u) \leq \mathrm{lfs}(w)/(1 - 0.3245)$. Hence $r \leq s \leq 0.3245\,\mathrm{lfs}(u) \leq \frac{0.3245}{0.6755}\,\mathrm{lfs}(v)$ and $r \leq \frac{0.3245}{0.6755}\,\mathrm{lfs}(w)$.

If τ's plane angle at v is 53.932° or greater, then by the Triangle Normal Lemma (Lemma 14), $\sin\angle(n_v, n_\tau) \leq r \cot 26.966°/\mathrm{lfs}(v) < (0.3245/0.6755)\cdot 1.9655 < 0.9442$. Therefore, $\angle(n_v, n_\tau) < 70.77°$ and $\angle(n_u, n_\tau) \leq \angle(n_v, n_u) + \angle(n_v, n_\tau) < 19.21° + 70.77° = 89.98°$.

Otherwise, τ's plane angle at v is less than 53.932°, so τ's largest plane angle (at w) is greater than $(180° - 53.932°)/2 = 63.034°$. By the Triangle Normal Lemma, $\sin\angle(n_w, n_\tau) \leq r \cot 31.517°/\mathrm{lfs}(w) < (0.3245/0.6755) \cdot 1.6308 < 0.78342$. Therefore, either $\angle(n_w, n_\tau) < 51.575°$ or $\angle(n_w, n_\tau) > 128.425°$. The latter case is not possible, because $\angle(n_w, n_\tau) \leq \angle(n_w, n_u) + \angle(n_u, n_v) + \angle(n_v, n_\tau) < 19.21° + 19.21° + 90° = 128.42°$. In the former case, $\angle(n_u, n_\tau) \leq \angle(n_w, n_u) + \angle(n_w, n_\tau) < 19.21° + 51.575° = 70.785°$ and $\angle(n_v, n_\tau) \leq \angle(n_v, n_u) + \angle(n_w, n_u) + \angle(n_w, n_\tau) < 19.21° + 19.21° + 51.575° = 89.995°$. ◀

▶ **Lemma 16.** *Let u be a restricted Voronoi vertex and let $\tau = \triangle vv'v''$ be its dual restricted Delaunay triangle. (If u's dual face is a polygon, you may choose any three of its vertices.) Let $w \in \{v, v', v''\}$ be the vertex at τ's largest plane angle. Let n_τ be a vector normal to τ, directed so that $\angle(n_v, n_\tau) \leq 90°$. Let $s = |vu| = |v'u| = |v''u| = |wu|$ and suppose that $s \leq 0.4132\,\mathrm{lfs}(v)$ and $s \leq 0.4132\,\mathrm{lfs}(w)$.*

Then $\angle(n_u, n_\tau) < 90°$ and $\angle(n_v, n_\tau) < 90°$. Equivalently, $n_u \cdot n_\tau$ and $n_v \cdot n_\tau$ are positive.

The proof of Lemma 16 is much like that of Lemma 15; see the full-length paper [32].

The final lemma shows that a nonempty intersection of two restricted Voronoi cells is either a lone restricted Voronoi vertex (a "degenerate" case where four or more restricted Voronoi cells share a restricted Voronoi vertex, whereas the common case is three cells sharing a vertex) or a lone topological interval justifying the name "restricted Voronoi edge."

▶ **Lemma 17.** *Let $\mathrm{Vor}|_\Sigma V$ be a restricted Voronoi diagram that satisfies all the conditions of Lemma 12, including the condition that at least three sites in V lie on each connected component of Σ. Suppose that closed ball property D and generic intersection properties E and F hold: no vertex of $\mathrm{Vor}\, V$ lies on Σ, and no edge and no 2-face of $\mathrm{Vor}\, V$ intersects Σ tangentially. Suppose also that for every restricted Voronoi vertex $u \in \mathrm{Vor}|_\Sigma V$, either*

- *$|vu| \leq 0.3245\,\mathrm{lfs}(u)$ for every site v such that $u \in \mathrm{Vor}|_\Sigma v$, or*
- *$|vu| \leq 0.4132\,\mathrm{lfs}(v)$ for every site v such that $u \in \mathrm{Vor}|_\Sigma v$.*

Let $v, w \in V$ be two distinct sites, let $F = \mathrm{Vor}\, v \cap \mathrm{Vor}\, w$, and let $f = \mathrm{Vor}|_\Sigma v \cap \mathrm{Vor}|_\Sigma w = F \cap \Sigma$.

Then one of these three claims holds: f is empty; f is a lone point (a restricted Voronoi vertex) and F is an edge; or f is homeomorphic to a closed interval and F is a 2-face.

Proof. Suppose f is nonempty. By Lemma 12, f is a union of disjoint topological intervals and isolated points. As f is nonempty and no vertex of $\mathrm{Vor}\, V$ lies on Σ (closed ball property D), F is not a vertex of $\mathrm{Vor}\, V$. Hence F is either an edge or a 2-face of $\mathrm{Vor}\, V$. If F is a Voronoi edge, then F is the intersection of three or more Voronoi cells and thus f is the intersection of three or more restricted Voronoi cells; so by assumption (the conditions of Lemma 12), f is a lone restricted Voronoi vertex and the result holds.

Only the case where F is a 2-face of $\mathrm{Vor}\, V$ remains. As closed ball property D and generic intersection properties E and F hold, $\Sigma \cap F$ cannot contain any isolated points; so f is a union of disjoint topological intervals. By Lemma 12, each interval contains exactly two restricted Voronoi vertices, its endpoints. Both of them lie on the boundary of F.

Let n_v be an outside-facing vector normal to Σ at v. We will see shortly that F is not perpendicular to n_v. Assign a direction to each edge of F such that the angle between n_v and each directed edge is at most 90°, as illustrated in Figure 6. As F is a convex polygon, we thus partition its edges into two chains, each monotone in the direction n_v. (An edge perpendicular to n_v – there are at most two – can be assigned to either chain.)

Let u be a restricted Voronoi vertex on an edge e of F. Let τ be u's dual restricted Delaunay triangle (or polygon) with vertices v and w. Let n_τ be a vector that points in the same direction as e, and observe that n_τ is normal to τ and $\angle(n_v, n_\tau) \leq 90°$. By Lemma 15 or Lemma 16, $\angle(n_u, n_\tau) < 90°$ and $\angle(n_v, n_\tau) < 90°$. The latter implies that F is not perpendicular to n_v (as promised). The former implies that as one walks along one of the directed monotone chains, one might encounter a restricted Voronoi vertex where the chain passes from inside Σ to outside Σ, but not from outside to inside. Therefore, there can be only one restricted Voronoi vertex on each chain, and only two restricted Voronoi vertices on F. It follows that $\mathrm{Vor}|_\Sigma v \cap \mathrm{Vor}|_\Sigma w$ is just a single topological interval. ◀

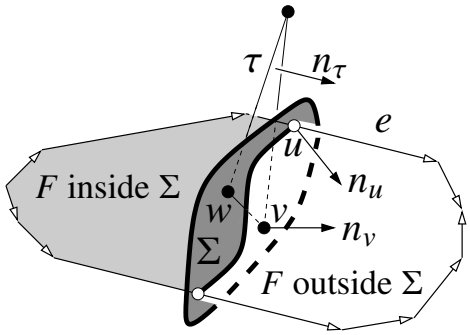


Figure 6 A Voronoi 2-face F, with its edges partitioned into two chains monotone in n_v.

7 The restricted Delaunay triangulation is homeomorphic to Σ

We conclude with homeomorphism theorems for 0.3245-samples and 0.4132-Voronoi samples.

▶ **Theorem 18.** *Let V be a finite ϵ-sample of Σ for some $\epsilon \leq 0.3245$. Suppose that no vertex of the three-dimensional Voronoi diagram $\mathrm{Vor}\, V$ lies on Σ. Then the underlying space of the restricted Delaunay triangulation, $|\mathrm{Del}|_\Sigma V|$, is homeomorphic to Σ.*

Proof. Corollary 9 guarantees closed ball property A and generic intersection property E. Corollary 11 guarantees closed ball property C and generic intersection property F. Closed ball property D holds by assumption. By Corollary 9, every connected component of Σ has at least six sites on it and no restricted Voronoi cell intersects the interior of another; all the preconditions of Lemmas 12 and 17 are satisfied. Lemma 17 guarantees closed ball property B. As Σ and V satisfy the preconditions A–F of the Topological Ball Theorem [23], $|\mathrm{Del}|_\Sigma V|$ is homeomorphic to Σ. ◀

▶ **Theorem 19.** *Let V be a finite ϵ-Voronoi sample of Σ for some $\epsilon \leq 0.4132$. Suppose that no vertex of the three-dimensional Voronoi diagram $\mathrm{Vor}\, V$ lies on Σ. Then the underlying space of the restricted Delaunay triangulation, $|\mathrm{Del}|_\Sigma V|$, is homeomorphic to Σ.*

Proof. Identical to the proof of Theorem 18, except that Corollary 9 is replaced by Theorem 6 and Lemmas 2, 7, and 5; and Corollary 11 is replaced by Lemma 10. ◀

References

1 Nina Amenta and Marshall W. Bern. Surface Reconstruction by Voronoi Filtering. *Discrete & Computational Geometry*, 22(4):481–504, June 1999. doi:10.1007/PL00009475.

2 Nina Amenta, Marshall W. Bern, and David Eppstein. The Crust and the β-Skeleton: Combinatorial Curve Reconstruction. *Graphical Models and Image Processing*, 60(2):125–135, March 1998. doi:10.1006/gmip.1998.0465.

3 Nina Amenta, Sunghee Choi, Tamal K. Dey, and Naveen Leekha. A Simple Algorithm for Homeomorphic Surface Reconstruction. *International Journal of Computational Geometry and Applications*, 12(1–2):125–141, 2002. doi:10.1142/S0218195902000773.

4 Dominique Attali and André Lieutier. Geometry-Driven Collapses for Converting a Čech Complex into a Triangulation of a Nicely Triangulable Shape. *Discrete & Computational Geometry*, 54(4):798–825, December 2015. doi:10.1007/s00454-015-9733-7.

5 Håvard Bakke Bjerkevik. Tighter Bounds for Reconstruction from ϵ-Samples. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:17, Berlin, Germany, June 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2022.9.

6 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and Topological Inference*, volume 57. Cambridge University Press, September 2018. doi:10.1017/9781108297806.

7 Jean-Daniel Boissonnat, Ramsay Dyer, and Arijit Ghosh. Delaunay Triangulation of Manifolds. *Foundations of Computational Mathematics*, 18(2):399–431, April 2018. doi:10.1007/s10208-017-9344-1.

8 Jean-Daniel Boissonnat, Ramsay Dyer, Arijit Ghosh, André Lieutier, and Mathijs Wintraecken. Local Conditions for Triangulating Submanifolds of Euclidean Space. *Discrete & Computational Geometry*, 66(2):666–686, September 2021. doi:10.1007/s00454-020-00233-9.

9 Jean-Daniel Boissonnat, Ramsay Dyer, Arijit Ghosh, and Nikolay Martynchuk. An Obstruction to Delaunay Triangulations in Riemannian Manifolds. *Discrete & Computational Geometry*, 59(1):226–237, January 2018. doi:10.1007/s00454-017-9908-5.

10 Jean-Daniel Boissonnat and Arijit Ghosh. Manifold Reconstruction Using Tangential Delaunay Complexes. *Discrete & Computational Geometry*, 51(1):221–267, January 2014. doi:10.1007/s00454-013-9557-2.

11 Jean-Daniel Boissonnat, Leonidas J. Guibas, and Steve Y. Oudot. Manifold Reconstruction in Arbitrary Dimensions Using Witness Complexes. *Discrete & Computational Geometry*, 42(1):37–70, July 2009. doi:10.1007/s00454-009-9175-1.

12 Jean-Daniel Boissonnat and Steve Oudot. Provably Good Surface Sampling and Approximation. In *Symposium on Geometry Processing*, pages 9–18. Eurographics Association, June 2003. doi:10.2312/SGP/SGP03/009-019.

13 Jean-Daniel Boissonnat and Steve Oudot. Provably Good Sampling and Meshing of Surfaces. *Graphical Models*, 67(5):405–451, September 2005. doi:10.1016/j.gmod.2005.01.004.

14 Dobrina Boltcheva and Bruno Lévy. Surface Reconstruction by Computing Restricted Voronoi Cells in Parallel. *Computer-Aided Design*, 90:123–134, September 2017. doi:10.1016/j.cad.2017.05.011.

15 Ho-Lun Cheng, Tamal Krishna Dey, Herbert Edelsbrunner, and John Sullivan. Dynamic Skin Triangulation. *Discrete & Computational Geometry*, 25(4):525–568, December 2001. doi:10.1007/s00454-001-0007-1.

16 Siu-Wing Cheng, Tamal Krishna Dey, and Edgar A. Ramos. Manifold Reconstruction from Point Samples. In *Proceedings of the Sixteenth Annual Symposium on Discrete Algorithms*, pages 1018–1027, Vancouver, British Columbia, Canada, January 2005. URL: http://dl.acm.org/citation.cfm?id=1070432.1070579.

17 Siu-Wing Cheng, Tamal Krishna Dey, Edgar A. Ramos, and Tathagata Ray. Sampling and Meshing a Surface with Guaranteed Topology and Geometry. *SIAM Journal on Computing*, 37(4):1199–1227, 2007. doi:10.1137/060665889.

18 Siu-Wing Cheng, Tamal Krishna Dey, and Jonathan Richard Shewchuk. *Delaunay Mesh Generation*. Chapman and Hall / CRC Computer and Information Science Series. CRC Press, Boca Raton, Florida, 2013. URL: http://www.crcpress.com/product/isbn/9781584887300.

19 David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Lexicographic Optimal Homologous Chains and Applications to Point Cloud Triangulations. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.SoCG.2020.32.

20 Tamal Krishna Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge University Press, New York, 2007. doi:10.1017/CBO9780511546860.

21 Ramsay Dyer, Hao Zhang, and Torsten Möller. Delaunay Mesh Construction. In *Proceedings of the Fifth Symposium on Geometry Processing*, pages 273–282. Eurographics Association, July 2007. doi:10.2312/SGP/SGP07/273-282.

22 Ramsay Dyer, Hao Zhang, and Torsten Möller. Surface Sampling and the Intrinsic Voronoi Diagram. *Computer Graphics Forum*, 27(5):1393–1402, July 2008. doi:10.1111/j.1467-8659.2008.01279.x.

23 Herbert Edelsbrunner and Nimish R. Shah. Triangulating Topological Spaces. *International Journal of Computational Geometry and Applications*, 7(4):365–378, August 1997. doi:10.1142/S0218195997000223.

24 Marc Khoury and Jonathan Richard Shewchuk. Fixed Points of the Restricted Delaunay Triangulation Operator. In Sándor P. Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:15, Boston, Massachusetts, June 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SOCG.2016.47.

25 Marc Khoury and Jonathan Richard Shewchuk. Approximation Bounds for Interpolation and Normals on Triangulated Surfaces and Manifolds, November 2019. arXiv:1911.03424.

26 Marc Khoury and Jonathan Richard Shewchuk. Restricted Constrained Delaunay Triangulations. In Kevin Buchin and Éric Colin de Verdière, editors, *37th International Symposium on Computational Geometry*, volume 189 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:16, Buffalo, New York, June 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2021.49.

27 Jisu Kim, Jaehyeok Shin, Frédéric Chazal, Alessandro Rinaldo, and Larry A. Wasserman. Homotopy Reconstruction via the Cech Complex and the Vietoris-Rips Complex. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.SoCG.2020.54.

28 Gregory Leibon. *Random Delaunay Triangulations, the Thurston–Andreev Theorem, and Metric Uniformization*. PhD thesis, University of California at San Diego, San Diego, California, 1999.

29 Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the Homology of Submanifolds with High Confidence from Random Samples. *Discrete & Computational Geometry*, 39(1–3):419–441, March 2008. doi:10.1007/s00454-008-9053-2.

30 Nicholas M. Patrikalakis and Takashi Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer, 2002. doi:10.1007/978-3-642-04074-0.

31 Jeanne Pellerin, Bruno Lévy, and Guillaume Caumon. Toward Mixed-Element Meshing Based on Restricted Voronoi Diagrams. *Procedia Engineering*, 82:279–290, 2014. doi:10.1016/j.proeng.2014.10.390.

32 Jonathan Richard Shewchuk. Better Sampling Bounds for Restricted Delaunay Triangulations and a Star-Shaped Property for Restricted Voronoi Cells, March 2026. arXiv:2603.19826.

33 Wilson A. Sutherland. *Introduction to Metric and Topological Spaces*. Oxford University Press, second edition, 2009. doi:10.1093/oso/9780199563074.001.0001.

34 Pengfei Wang, Zixiong Wang, Shiqing Xin, Xifeng Gao, Wenping Wang, and Changhe Tu. Restricted Delaunay Triangulation for Explicit Surface Reconstruction. *ACM Transactions on Graphics*, 41(5):180:1–180:20, October 2022. doi:10.1145/3533768.

35 Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. Isotropic Remeshing with Fast and Exact Computation of Restricted Voronoi Diagram. *Computer Graphics Forum*, 28(5):1445–1454, 2009. doi:10.1111/j.1467-8659.2009.01521.x.

The Hierarchy of Manifolds in a Stratification of the Set of Equivalent Linear Neural Networks

Jonathan Richard Shewchuk
University of California, Berkeley, CA, USA

Sagnik Bhattacharya
University of California, Berkeley, CA, USA

Abstract

A linear neural network computes a linear transformation of its input vector. Given a fully-connected linear network, the set of all weight vectors for which the network computes the same linear transformation is an algebraic variety in weight space, called a *fiber* under the *matrix multiplication map*. Sometimes this variety is a manifold, but usually not. The *rank stratification* of a fiber is a natural partition of the fiber into manifolds of various dimensions called *strata*. We characterize how these strata are connected to each other. They satisfy the *frontier condition*: if a stratum intersects the closure of another stratum, then the former stratum is a subset of the closure of the latter stratum. This subset relationship can be expressed as a partial order with a single minimal element. Our main result describes the relationship between this partial order and the ranks of certain matrices in the network. Each stratum represents a different pattern of information flow through the network, expressed as a *barcode*. Connections among the strata are best understood through simple transformations of the barcodes called *barcode moves*.

2012 ACM Subject Classification Computing methodologies → Neural networks; Mathematics of computing → Nonlinear equations; Theory of computation → Computational geometry

Keywords and phrases Linear neural network, real algebraic variety, stratification, multilinear algebra, product of matrices, persistence barcode, real algebraic geometry, discrete geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.91

Related Version *Full Version*: `https://arxiv.org/abs/2404.14855` [20]

Funding Supported in part by the National Science Foundation under Award CCF-1909204.

Acknowledgements We thank Marc Khoury for initiating this research collaboration, and we thank Herbert Edelsbrunner, Jeff Erickson, Marc Glisse, and Ling Zhou for conversations helping us to understand the relationship between this work and persistence modules.

1 Introduction

In its simplest form, a *linear neural network* is a sequence of matrices whose product is a matrix.[1] The network takes an *input vector* x and produces an *output vector* $y = W_L W_{L-1} \cdots W_2 W_1 x$. We number the matrices in the order they are applied. The matrix W_1 linearly transforms x, producing a vector that W_2 linearly transforms, and so on. The composition of those transformations is also a linear transformation, represented by

$$W = \mu(W_L, W_{L-1}, \ldots, W_2, W_1) = W_L W_{L-1} \cdots W_2 W_1,$$

where μ is called the *matrix multiplication map*. Each matrix W_j is interpreted as a layer of edges (connections) in the network, edge layer number j, and each component of W_j is interpreted as the *weight* of an edge. For brevity, we omit added terms, which do not appreciably affect our results (but we address them in the full-length manuscript [20]).

[1] To a practitioner, a linear neural network is a neural network in which all the activation functions are the identity function. In this paper's neural networks, every layer of edges is fully-connected.

© Jonathan Richard Shewchuk and Sagnik Bhattacharya;
licensed under Creative Commons License CC-BY 4.0

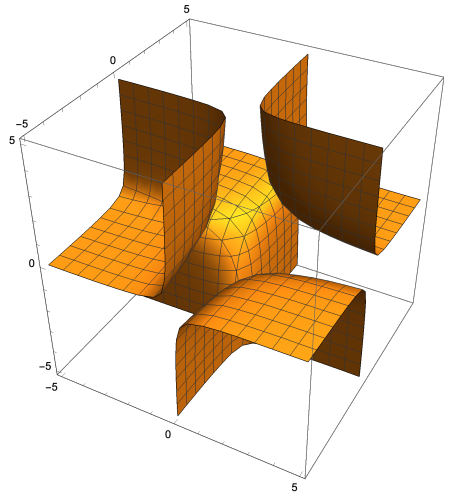


Figure 1 The fiber $\mu^{-1}([1])$ for the network $W_3W_2W_1 = [\theta_3][\theta_2][\theta_1] = [1] = W$.

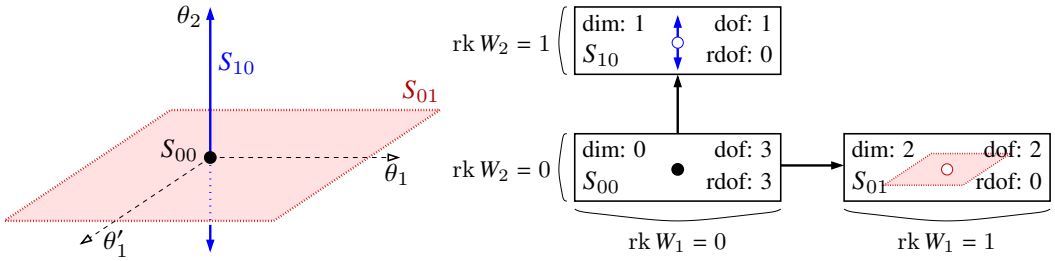


Figure 2 At left is the fiber $\mu^{-1}([0 \quad 0])$ for the network $W_2W_1 = [\theta_2][\theta_1 \quad \theta_1'] = [0 \quad 0] = W$, partitioned into three strata: S_{00} is the origin; S_{10} (blue) is the θ_2-axis with the origin removed; and S_{01} (pink) is the plane spanned by the θ_1- and θ_1'-axes with the origin removed. At right, the strata are arranged in a *stratum dag*, organized as a table indexed by the ranks of W_1 and W_2. Each dag vertex specifies the dimension of the stratum (dim), the number of degrees of freedom of motion on the fiber (dof = dim + rdof), and the number of rank-increasing degrees of freedom (rdof), which stay on the fiber but move off the stratum and onto a higher-dimensional stratum. A directed path from one stratum to another implies that the former is a subset of the closure of the latter.

We wish to study $\mu^{-1}(W)$, the set of all factorizations of a matrix W into a product of matrices of specified sizes. Said differently, we study the set of all choices of linear neural network weights such that the network computes the linear transformation W. This set is infinite (unless $L = 1$) and it is a *real algebraic variety* – the set of all real-valued solutions of a system of polynomial equations (specifically, multilinear equations). Trager, Kohn, and Bruna [23], whose paper inspired ours, call $\mu^{-1}(W)$ the *fiber* of W under the map μ. Understanding the fiber has applications in understanding gradient descent algorithms for training neural networks – but it is also a beautiful mathematical problem in its own right.

Sometimes the fiber is a manifold – see Figure 1 for an example – but usually not. Always it can be partitioned into smooth manifolds of various dimensions, called *strata*. The set of strata is called a *stratification* of the fiber. Figure 2 graphs the solutions of $[\theta_2][\theta_1 \quad \theta_1'] = [0 \quad 0]$ (an instantiation of $W_2W_1 = W$). The set of solutions can be partitioned into three strata: S_{00} is the origin, S_{10} is the θ_2-axis with the origin removed, and S_{01} is the θ_1-θ_1' plane with the origin removed. The two subscripts of S_{**} are the ranks of W_2 and W_1, respectively. The stratum S_{00} lies in the closures of both S_{10} and S_{01}; S_{00} serves to connect S_{10} to S_{01}.

Figure 2 illustrates what we call a *rank stratification* of the fiber $\mu^{-1}(W)$, which has one stratum for each *rank list* defined in Section 2; it is the most natural way to stratify a fiber of the matrix multiplication map. In the full-length manuscript [20], we show that the strata in a rank stratification are analytic manifolds (hence of class C^∞), derive their dimensions, and derive the tangent and normal spaces at each point of each stratum.

This paper characterizes how the strata are connected to each other. The strata satisfy the *frontier condition*: if a stratum intersects the closure of another stratum, then the former stratum is a subset of the closure of the latter stratum. This subset relationship induces a partial order with a single minimal element, a stratum that is included in the closure of every other stratum (in Figure 2, S_{00}). Our main result describes the relationship between this partial order and the ranks of certain matrices in the network. What makes it interesting is that each stratum represents a different pattern of information flow through the network.

In 1957, Hassler Whitney [24, 25, 26] proved that every real algebraic variety can be partitioned into a finite set of analytic manifolds. Łojasiewicz [16] generalized the result to semi-algebraic and semi-analytic sets. Mather [18] showed that Whitney's stratifications satisfy the frontier condition (but there exist stratifications that do not, hence we prove that rank stratifications do). Thom [22] introduced the terms *stratum* and *stratification*.

Linear neural networks compute only linear transformations; they are far less powerful than networks with nonlinear activation functions such as rectified linear units (ReLUs, also called ramp functions) and sigmoid functions (also called logistic functions). Yet linear networks have become a popular object of study [1, 2, 4, 5, 11, 12, 13, 14, 17, 19, 23, 27]. Why? We cannot fully understand the training of ReLU-based networks – or probably any neural networks – if we do not understand linear networks. Researchers have studied linear neural networks to understand phenomena observed in ReLU networks such as implicit regularization in optimizing the training algorithm's cost function [10, 2, 5], implicit acceleration of training by gradient descent [1], and the success of residual networks [11]. Close to our hearts, Trager, Kohn, and Bruna [23] show that the fiber $\mu^{-1}(W)$ is crucial for understanding the critical points of the cost functions minimized by neural network training algorithms. We aspire to build a related theory for ReLU networks, and we hope insights in this paper will help.

2 Notation

Let L be the number of matrices – that is, the number of layers of edges (connections) in the network. Alternating with the edge layers are $L+1$ layers of units, numbered from 0 to L, in which layer j has d_j real-valued *units* that represent a vector in $\mathbb{R}^{d_j}$. Unit layer 0 is the *input layer*, unit layer L is the *output layer*, and between them are $L-1$ *hidden layers*. The layers of edges are numbered from 1 to L, and the edge weights in edge layer j are represented by a real-valued $d_j \times d_{j-1}$ matrix W_j.

We collect all the network's weights in a *weight vector* $\theta = (W_L, W_{L-1}, \ldots, W_1) \in \mathbb{R}^{d_\theta}$, where $d_\theta = d_L d_{L-1} + d_{L-1} d_{L-2} + \ldots + d_1 d_0$ is the number of real-valued weights in the network (i.e., the number of connections). Recall the matrix multiplication map $\mu(W_L, W_{L-1}, \ldots, W_2, W_1)$; we can abbreviate it to $\mu(\theta) = W_L W_{L-1} \cdots W_2 W_1$. Given a fixed weight vector θ, the linear neural network takes an *input vector* $x \in \mathbb{R}^{d_0}$ and returns an *output vector* $y = W_L W_{L-1} \cdots W_2 W_1 x$, with $y \in \mathbb{R}^{d_L}$. Hence, the network implicitly applies a linear transformation specified by the $d_L \times d_0$ matrix $W = \mu(\theta)$, yielding $y = Wx$.

The map μ is not bijective (unless $L = 1$), so it has a preimage set, not an inverse. Given $W \in \mathbb{R}^{d_L \times d_0}$, let

$$\mu^{-1}(W) = \{\theta : \mu(\theta) = W\}$$

be the set of all factorizations of W for some fixed $d_L, d_{L-1}, \ldots, d_0$. Call $\mu^{-1}(W)$ the *fiber* of W under μ; it is a real algebraic variety in the weight space $\mathbb{R}^{d_\theta}$. With respect to $\mathbb{R}^{d_\theta}$, $\mu^{-1}(W)$ is a closed point set. Note that $\mu^{-1}(W)$ is empty if and only if $\operatorname{rk} W > \min_{j=1}^{L-1} d_j$.

Given a weight vector $\theta = (W_L, W_{L-1}, \ldots, W_1)$, its *subsequence matrices* are all the matrices of the form $W_{k \blacktriangleleft i} = W_k W_{k-1} \cdots W_{i+1}$. The notation $W_{k \blacktriangleleft i}$ indicates that this matrix transforms a vector at unit layer i to produce a vector at unit layer k. Note that $W = W_{L \blacktriangleleft 0}$ and $W_j = W_{j \blacktriangleleft j-1}$. We use the convention that $W_{k \blacktriangleleft k} = I_{d_k \times d_k}$, the $d_k \times d_k$ identity matrix. Assume that every $W_{k \blacktriangleleft i}$ in this paper is a function of θ. We call each W_j a *factor matrix*.

The *rank list* $\underline{r}$ for a weight vector $\theta \in \mathbb{R}^{d_\theta}$ is a sequence that lists the rank of every subsequence matrix $W_{k \blacktriangleleft i}$ such that $L \geq k \geq i \geq 0$. The list includes the unit layer sizes $\operatorname{rk} W_{k \blacktriangleleft k} = d_k$. For example, for a network with $L = 3$ layers of edges, the rank list of θ is

$$\underline{r} = \langle d_3, d_2, d_1, d_0, \operatorname{rk} W_3, \operatorname{rk} W_2, \operatorname{rk} W_1, \operatorname{rk} W_3 W_2, \operatorname{rk} W_2 W_1, \operatorname{rk} W \rangle.$$

We will use rank lists to partition a fiber into strata. Sometimes we do not want to specify a particular θ, but rather we wish to specify some target ranks. In that case, we let $r_{k \blacktriangleleft i}$ denote the target value of $\operatorname{rk} W_{k \blacktriangleleft i}$ and we write $\underline{r} = \langle r_{k \blacktriangleleft i} \rangle_{L \geq k \geq i \geq 0}$. For example, if we set $r_{3 \blacktriangleleft 1} = 2$, we select weight vectors for which $\operatorname{rk} W_3 W_2 = 2$. If $L = 2$ and we set $\underline{r} = \langle 9, 8, 7, 2, 3, 1 \rangle$, we select weight vectors for which W_2 is a 9×8 matrix of rank 2, W_1 is an 8×7 matrix of rank 3, and $W = W_2 W_1$ has rank 1.

Let $S_{\underline{r}}^W$ denote the set of points in W's fiber having rank list $\underline{r}$. That is,

$$S_{\underline{r}}^W = \{\theta \in \mu^{-1}(W) : \text{the rank list of } \theta \text{ is } \underline{r}\}.$$

If it is nonempty, we call $S_{\underline{r}}^W$ a *stratum* in the rank stratification of W's fiber. When W is clear from context, we just write $S_{\underline{r}}$. Each $S_{\underline{r}}$ is an analytic manifold [20] (but not necessarily closed nor connected nor bounded). The *rank stratification* of $\mu^{-1}(W)$ is

$$\mathbb{S} = \{S_{\underline{r}} : \underline{r} \text{ is the rank list of some weight vector in } \mu^{-1}(W)\}.$$

The members of $\mathbb{S}$ are disjoint and $\mu^{-1}(W) = \bigcup_{S \in \mathbb{S}} S$ – that is, $\mathbb{S}$ is a partition of $\mu^{-1}(W)$.

A rank list $\underline{r}$ is *valid* if there exists some weight vector $\theta \in \mathbb{R}^{d_\theta}$ that has rank list $\underline{r}$. We show in the full-length manuscript [20] that if $W \in \mathbb{R}^{d_L \times d_0}$ and $\underline{r}$ is a valid rank list with $r_{L \blacktriangleleft 0} = \operatorname{rk} W$, then there exists some $\theta \in \mu^{-1}(W)$ that has rank list $\underline{r}$, so $S_{\underline{r}}$ is nonempty.

Given two rank lists $\underline{r}$ and $\underline{z}$, we write $\underline{r} \leq \underline{z}$ to mean that $r_{k \blacktriangleleft i} \leq z_{k \blacktriangleleft i}$ for all $L \geq k \geq i \geq 0$. We write $\underline{r} < \underline{z}$ to mean that $\underline{r} \leq \underline{z}$ and $\underline{r} \neq \underline{z}$ (at least one of the inequalities holds strictly).

Given a set $S \subseteq \mathbb{R}^{d_\theta}$, let $\bar{S}$ denote the closure of S (with respect to the weight space $\mathbb{R}^{d_\theta}$).

3 Main result: The hierarchy of strata

▶ **Theorem 1.** *Let $\underline{r}$ and $\underline{z}$ be two valid rank lists for the same linear neural network (i.e., $r_{j \blacktriangleleft j} = z_{j \blacktriangleleft j} = d_j$ for all $j \in [0, L]$). Let $\mu^{-1}(W)$ be the fiber of a matrix $W \in \mathbb{R}^{d_L \times d_0}$ whose rank satisfies $\operatorname{rk} W = r_{L \blacktriangleleft 0} = z_{L \blacktriangleleft 0}$. Let $S_{\underline{r}}$ and $S_{\underline{z}}$ be the strata with rank lists $\underline{r}$ and $\underline{z}$ in the rank stratification $\mathbb{S}$ of $\mu^{-1}(W)$, and recall that validity implies that $S_{\underline{r}} \neq \emptyset$ and $S_{\underline{z}} \neq \emptyset$. Then the following statements are equivalent (imply each other).*

A. $S_{\underline{r}} \subseteq \bar{S}_{\underline{z}}$.
B. $S_{\underline{r}} \cap \bar{S}_{\underline{z}} \neq \emptyset$.
C. $\underline{r} \leq \underline{z}$.
D. *There exists a sequence of rank-one abstract barcode moves that proceed from $\underline{r}$ to $\underline{z}$, with all the intermediate rank lists being valid.*

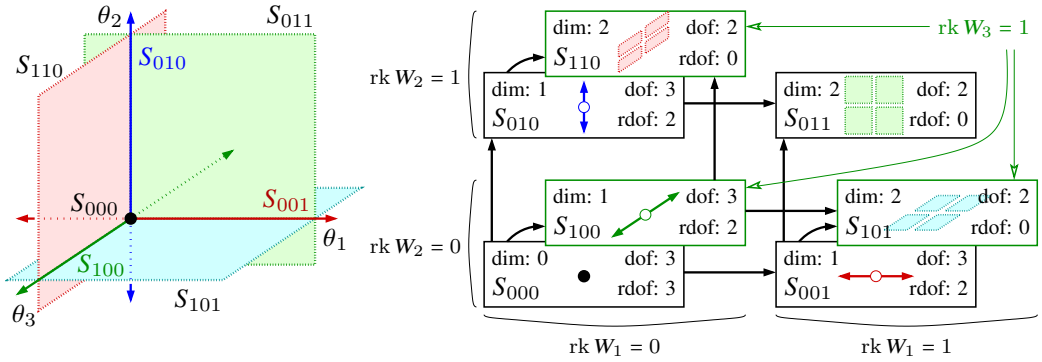


Figure 3 At left is the variety of solutions to $W_3W_2W_1 = [\theta_3][\theta_2][\theta_1] = [0] = W$, partitioned into seven strata: S_{000} is the origin; S_{001}, S_{010}, and S_{100} are the three coordinate axes with the origin removed; and S_{011}, S_{101}, and S_{110} are the three coordinate planes with the coordinate axes removed. At right is the stratum dag, organized as a table indexed by the ranks of W_3, W_2, and W_1.

We have not yet defined the terminology in statement D. A *stratum dag* is a directed acyclic graph (dag) with one vertex for each stratum in $\mathbb{S}$ and one directed edge for each rank-one abstract barcode move; see Figure 2, right. For now, just know that these moves are the central new concept in this paper, which organize the strata in $\mathbb{S}$ in a natural hierarchy. Read statement D as "there is a directed path in the stratum dag from $S_{\underline{r}}$ to $S_{\underline{z}}$." The stratum dag encodes the partial order of the strata induced by $\leq$, but it is not necessarily the simplest dag that does so (sometimes called a *Hasse diagram* or *transitive reduction*), for reasons we explain in the full-length manuscript [20].

Figures 3 and 4 depict two more examples of stratum dags. Notable in Figure 3 is that we arrange the stratum dag in a three-dimensional table, indexed by $\operatorname{rk} W_3$, $\operatorname{rk} W_2$, and $\operatorname{rk} W_1$. Ordinarily, three-matrix fibers ($L = 3$) require five indices to index the strata, as $\operatorname{rk} W_3W_2$ and $\operatorname{rk} W_2W_1$ can vary as well; but in this example every matrix is 1×1, so those two ranks are uniquely determined by the first three. Notable in Figure 4 is that the fiber has dimension as high as 35 at some points, and it is embedded in a 54-dimensional weight space. Unfortunately, we cannot visualize the fiber, but its strata are curved, like in Figure 1. In both dags, there is a unique minimal stratum – S_{000} and S_{11}, respectively – that is a subset of the closure of every other stratum. In all rank stratifications, the minimal stratum is the one in which every subsequence matrix's rank is $\operatorname{rk} W$, except the layer sizes $\operatorname{rk} W_{j \blacktriangleleft j} = d_j$. (In the full-length manuscript [20], we give an algorithm for computing stratum dags.)

Most important in Theorem 1 is the equivalence of statements A and C: $S_{\underline{r}} \subseteq \bar{S}_{\underline{z}}$ if and only if $\underline{r} \leq \underline{z}$. The rank lists tell you which strata's closures include (or intersect) which strata.

The fact that B implies A means that rank stratifications satisfy the frontier condition defined in the Introduction: for every pair of strata $S, T \in \mathbb{S}$, either $S \cap \bar{T} = \emptyset$ or $S \subseteq \bar{T}$. (The two possibilities are mutually exclusive, because $S \neq \emptyset$.) That is, the inclusion of points of S in $\bar{T}$ is all or nothing, which helps us to understand how strata are connected to each other. For example, if $S \subseteq \bar{T}$ and $S \neq T$, then $\dim S < \dim T$ and from any point on S there is an infinitesimal perturbation that takes us onto T.

As the fiber is a closed point set, Theorem 1 implies that $\bar{S}_{\underline{z}} = \bigcup_{\underline{r} \leq \underline{z}} S_{\underline{r}}$. In Figure 3, $\bar{S}_{011} = S_{011} \cup S_{010} \cup S_{001} \cup S_{000}$, whereas $S_{100} \cap \bar{S}_{011} = \emptyset$ and $S_{011} \cap \bar{S}_{100} = \emptyset$.

Our proof of Theorem 1 shows the four implications A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D $\rightarrow$ A. Clearly A implies B, as we assume $S_{\underline{r}} \neq \emptyset$. To see that B implies C, observe that an infinitesimal perturbation of a weight vector $\theta \in S_{\underline{r}}$ may increase the ranks of some subsequence matrices

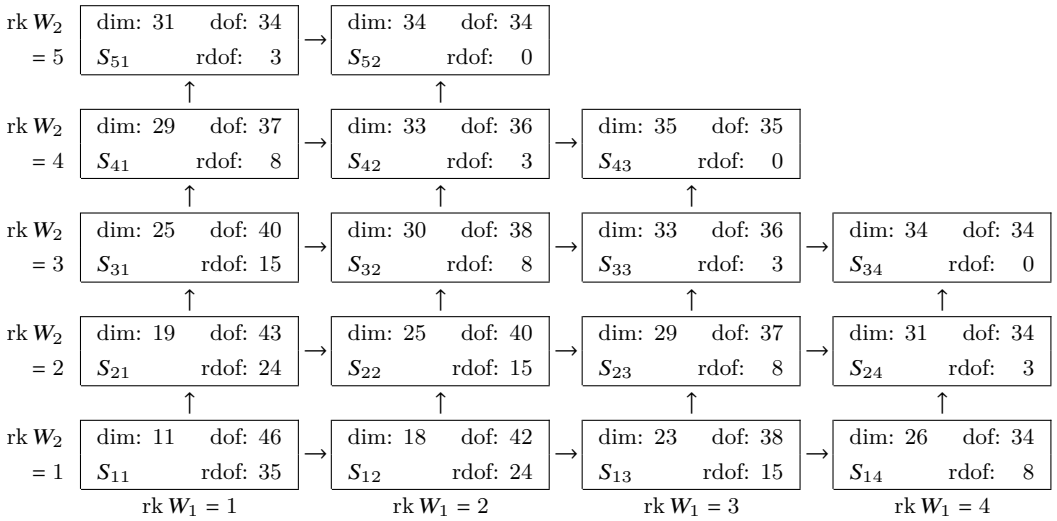


Figure 4 Stratum dag representing the stratification of $\mu^{-1}(W)$ for $W = W_2W_1$, $W_2 \in \mathbb{R}^{5\times 6}$, $W_1 \in \mathbb{R}^{6\times 4}$, and $\operatorname{rk} W = 1$. For every pair of strata S_{ts} and $S_{t's'}$ with $t \le t'$ and $s \le s'$, $S_{ts} \subseteq \bar{S}_{t's'}$.

but it cannot decrease any of their ranks, so a stratum $S_{\underline{z}}$ can have θ in its closure only if $\underline{r} \le \underline{z}$. Showing that D implies A is not difficult (an abstract move implies arbitrarily small geometric moves). The fact that C implies D was by far the hardest of the four implications to prove; for many months we did not know if C implies any of the other statements.

While statements A and B are statements about geometry, statements C and D are purely combinatorial. Our proof that C implies D is also purely combinatorial. Space is limited, so we give only part of that proof, including an algorithm that finds a suitable sequence of barcode moves (thereby showing one exists). The rest of this paper relates the geometry to the combinatorics, introduces barcode moves, and constructs the sequence of moves.

4 Prebases, fundamental subspaces, and barcodes

In this section, we show how to decompose each unit layer's space $\mathbb{R}^{d_j}$ into a *prebasis* of subspaces of $\mathbb{R}^{d_j}$; then decompose each edge layer's space $\mathbb{R}^{d_j\times d_{j-1}}$ into a prebasis of subspaces of $\mathbb{R}^{d_j\times d_{j-1}}$; then decompose the weight space $\mathbb{R}^{d_\theta}$ into a prebasis of subspaces of $\mathbb{R}^{d_\theta}$, which will illuminate the connections among strata. We assume the reader is familiar with the standard idea from linear algebra of a *basis* for $\mathbb{R}^d$, comprising d linearly independent *basis vectors*. A prebasis is like a basis, but it is made up of subspaces rather than vectors.

Given two subspaces $P, Q \subseteq \mathbb{R}^d$, their vector sum is $P + Q = \{p + q : p \in P \text{ and } q \in Q\}$. If P and Q are linearly independent – that is, if $P \cap Q = \{\mathbf{0}\}$ – then $P + Q$ is called a *direct sum*, sometimes written[2] $P \oplus Q$. Likewise, given a set of subspaces $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$, the direct sum notation $P_1 \oplus P_2 \oplus \ldots \oplus P_m$ implies that the subspaces in $\mathcal{P}$ are linearly independent – meaning that for every $i \in [1, m]$, $P_i \cap \sum_{j\neq i} P_j = \{\mathbf{0}\}$.

If $\mathbb{R}^d = P_1 \oplus P_2 \oplus \ldots \oplus P_m$, then $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ is known as a *direct sum decomposition* of $\mathbb{R}^d$. For brevity, we call $\mathcal{P}$ a *prebasis* for $\mathbb{R}^d$, and we say that $\mathcal{P}$ *spans* $\mathbb{R}^d$. We call each P_i a *prebasis subspace* – a multidimensional analogue of a basis vector. The linear independence

[2] The notation $P \oplus Q$ is unconventional, because as an operator it produces $P + Q$, but it also implies a constraint on the subspaces P and Q: that $P \cap Q = \{\mathbf{0}\}$. If $P \cap Q \neq \{\mathbf{0}\}$, then $P \oplus Q$ is undefined.

of the prebasis subspaces implies that for every vector $v \in \mathbb{R}^d$, there is one and only one way to express v as a sum of vectors $v = \sum_{i=1}^{m} v_i$ such that $v_i \in P_i$. It also implies that $d = \sum_{i=1}^{m} \dim P_i$. If desired, it is conceptually easy to convert a prebasis into a traditional vector basis: just choose a basis for each P_i, then pool the d vectors together to form a basis for $\mathbb{R}^d$ – hence the name "prebasis." Why don't we do that here? Because details like the choice of basis for each prebasis subspace and the length of each basis vector are irrelevant and would make our presentation more complicated.

The following theorem identifies "fundamental" subspaces in a linear neural network's hidden layers analogous to the four fundamental subspaces [21] of a matrix M: the rowspace (denoted $\operatorname{row} M$), the nullspace ($\operatorname{null} M$), the columnspace ($\operatorname{col} M$), and the left nullspace ($\operatorname{null} M^\top$). The theorem originates in the study of persistent homology [8, 7] (for entirely different reasons) where it is used to analyze persistence modules [28, 29, 9, 3, 15], which are essentially linear neural networks – but usually over the field $\mathbb{Z}/2\mathbb{Z}$, in contrast to our real-valued weights. Given a matrix M and a subspace $P \subseteq \mathbb{R}^d$, define $MP = \{Mv : v \in P\}$.

▶ **Theorem 2** (Fundamental Theorem of Linear Neural Networks). *Let $\theta = (W_L, W_{L-1}, \ldots, W_1)$ be a weight vector representing a linear neural network with L layers of edges and unit layer sizes $d_L, d_{L-1}, \ldots, d_0$. For all indices k and i satisfying $L \geq k \geq i \geq 0$, let*

$$\omega_{ki} = \operatorname{rk} W_{k \blacktriangleleft i} - \operatorname{rk} W_{k \blacktriangleleft i-1} - \operatorname{rk} W_{k+1 \blacktriangleleft i} + \operatorname{rk} W_{k+1 \blacktriangleleft i-1}, \tag{1}$$

using the conventions that $\operatorname{rk} W_{j \blacktriangleleft j} = d_j$, $\operatorname{rk} W_{L+1 \blacktriangleleft i} = 0$, and $\operatorname{rk} W_{k \blacktriangleleft -1} = 0$.

Then there exist subspaces $a_{kji} \subseteq \mathbb{R}^{d_j}$ and $b_{kji} \subseteq \mathbb{R}^{d_j}$ of dimension ω_{ki} for all indices satisfying $L \geq k \geq j \geq i \geq 0$, such that for all $j \in [0, L]$,

- *$\mathcal{A}_j = \{a_{kji} \neq \{\mathbf{0}\} : k \in [j, L], i \in [0, j]\}$ is a direct sum decomposition of $\mathbb{R}^{d_j}$ and*
- *$\mathcal{B}_j = \{b_{kji} \neq \{\mathbf{0}\} : k \in [j, L], i \in [0, j]\}$ is also a direct sum decomposition of $\mathbb{R}^{d_j}$;*

and the subspaces satisfy the flow relationships

$$W_j a_{k,j-1,i} = \begin{cases} a_{kji}, & k \geq j, \\ \{\mathbf{0}\}, & k = j-1, \end{cases} \qquad \forall j \in [1, L], k \in [j-1, L], i \in [0, j-1], \qquad \text{and}$$

$$W_j^\top b_{kji} = \begin{cases} b_{k,j-1,i}, & j > i, \\ \{\mathbf{0}\}, & j = i, \end{cases} \qquad \forall j \in [1, L], k \in [j, L], i \in [0, j].$$

This decomposition into subspaces a_{kji} (or b_{kji}) is not necessarily unique, but the dimensions $\dim a_{kji} = \dim b_{kji} = \omega_{ki}$ are the same for all such decompositions. Moreover,

$$d_j = \sum_{t=j}^{L} \sum_{s=0}^{j} \omega_{ts}, \quad \forall j \in [0, L], \qquad \text{and} \qquad \operatorname{rk} W_{k \blacktriangleleft i} = \sum_{t=k}^{L} \sum_{s=0}^{i} \omega_{ts}, \quad L \geq k \geq i \geq 0. \tag{2}$$

The prebases $\mathcal{A}_j$ describe information flow through the network. As illustrated in Figure 5 (top), we interpret a_{kii} as an ω_{ki}-dimensional subspace that appears at unit layer i (its *birth layer*), being linearly independent of (but not necessarily orthogonal to) $\operatorname{col} W_i$, then flows through the weight layers $W_{i+1}, W_{i+2}, \ldots, W_k$ being linearly transformed into a sequence of subspaces $a_{k,i+1,i}, a_{k,i+2,i}, \ldots$, all of dimension ω_{ki}, reaches layer k (its *death layer*) in the form a_{kki}, still having ω_{ki} dimensions, and proceeds no farther – because either $a_{kki} \subseteq \operatorname{null} W_{k+1}$ or layer k is the output layer. These subspaces carry information from the input layer only if $i = 0$ and reach the output layer only if $k = L$; otherwise, they represent unused potential.

Symmetrically, the prebases $\mathcal{B}_j$ govern the transpose network $W^\top = W_1^\top W_2^\top \cdots W_{L-1}^\top W_L^\top$. We interpret b_{kki} as an ω_{ki}-dimensional subspace that appears at layer k, being linearly independent of $\operatorname{row} W_{k+1}$, then flows through the weight layers $W_k^\top, W_{k-1}^\top, \ldots, W_{i+1}^\top$ being linearly transformed into a sequence of ω_{ki}-dimensional subspaces $b_{k,k-1,i}, b_{k,k-2,i}, \ldots$, and stops at layer i in the form b_{kii} – either because $b_{kii} \subseteq \operatorname{null} W_i^\top$ or because $i = 0$.

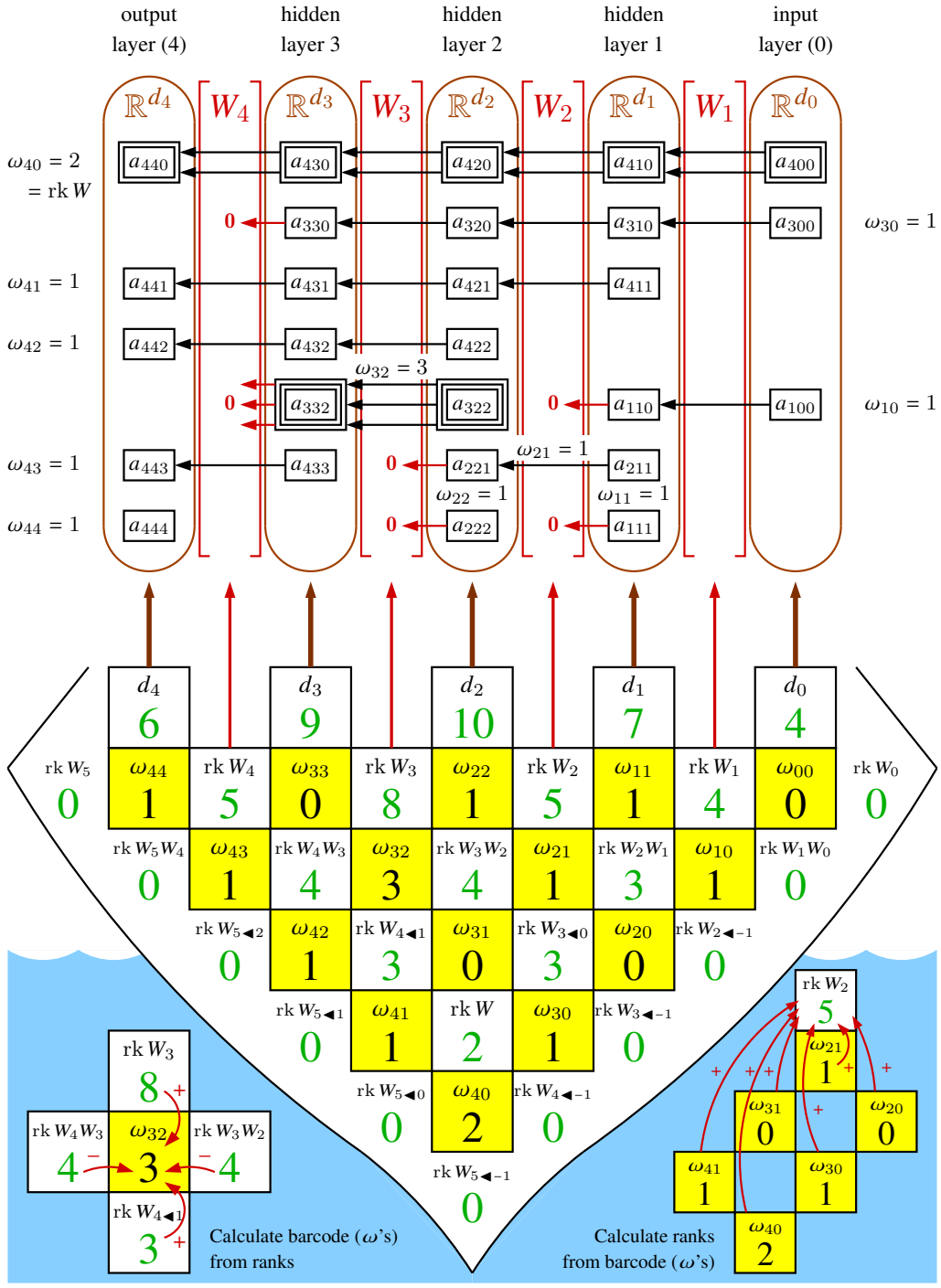


Figure 5 The top half is an annotated barcode that illustrates the flow of the prebasis subspaces a_{kji} through the network. Double boxes represent subspaces of dimension 2 and triple boxes represent subspaces of dimension 3. The bottom half relates the interval counts, the layer sizes, and the subsequence matrix ranks. The number of units d_j in unit layer j is the sum of the counts ω_{ts} of the intervals that touch layer j (i.e., the dimensions of the subspaces a_{tjs}). Each subsequence matrix rank $\operatorname{rk} W_{k\blacktriangleleft i}$ is the sum of the counts of the intervals that touch both layers k and i.

We call each ω_{ki} an *interval count*. An *interval* is a set of consecutive integers $[i,k] = \{i, i+1, \ldots, k-1, k\}$ that signifies consecutive unit layers, with i being the birth layer and k being the death layer for the information that the interval represents. The *barcode* for θ – an idea originating in persistent homology [9] – is a multiset of intervals that contains ω_{ki} copies of $[i,k]$ for all $0 \le i \le k \le L$. The top half of Figure 5 depicts a barcode annotated with prebasis subspaces. Barcodes show visually how information flows in linear networks.

The identities (1) and (2) express a bijection between rank lists and barcodes. The bottom half of Figure 5 depicts those identities visually.

We will see in Section 6 that we can understand the hierarchy of strata in terms of simple transformations of the barcode. (We can also understand it in terms of transformations of the rank list, but those transformations are not as simple.)

Now, we construct prebases for the factor matrix spaces $\mathbb{R}^{d_j \times d_{j-1}}$ and the weight space $\mathbb{R}^{d_\theta}$. Like the a_{kji}'s and b_{kji}'s, these prebases are defined at a specific weight vector θ. For indices satisfying $L \ge l \ge j \ge i \ge 0$ and $L \ge k \ge j-1 \ge h \ge 0$, define the prebasis subspace

$$o_{lkjih} = a_{lji} \otimes b_{k,j-1,h} = \{M \subseteq \mathbb{R}^{d_j \times d_{j-1}} : \operatorname{col} M \subseteq a_{lji} \text{ and } \operatorname{row} M \subseteq b_{k,j-1,h}\}.$$

Then $\dim o_{lkjih} = \dim a_{lji} \cdot \dim b_{k,j-1,h} = \omega_{li}\,\omega_{kh}$. For each $j \in [1, L]$, define the prebasis

$$O_j = \{o_{lkjih} \ne \{0\} : l \in [j, L], k \in [j-1, L], i \in [0, j], h \in [0, j-1]\}.$$

O_j is a prebasis for the weight layer $\mathbb{R}^{d_j \times d_{j-1}}$, as it pairs every subspace in the prebasis $\mathcal{A}_j$ (spanning $\mathbb{R}^{d_j}$) with every subspace in the prebasis $\mathcal{B}_{j-1}$ (spanning $\mathbb{R}^{d_{j-1}}$).

Now we construct a prebasis Θ_{O} for the weight space $\mathbb{R}^{d_\theta}$ that we call the *one-matrix prebasis*. The subspaces in Θ_{O} are called the *one-matrix subspaces* and have the form

$$\phi_{lkjih} = \{(0, \ldots, 0, M, 0, \ldots, 0) : M \in o_{lkjih}\}$$

with M in position j from the right (the position of W_j in θ). The one-matrix prebasis is

$$\Theta_{\mathrm{O}} = \{\phi_{lkjih} \ne \{\mathbf{0}\} : L \ge l \ge j \ge i \ge 0 \text{ and } L \ge k \ge j-1 \ge h \ge 0\}.$$

5 Moves on (and off) the fiber and the stratum

Imagine you are standing at a point θ on a fiber $\mu^{-1}(W)$. A *move* (θ, θ') is a step you take from θ to another point θ', which may or may not be on the fiber. Let $\Delta\theta = \theta' - \theta$ be the *displacement* of the move. We write

$$\theta' = (W'_L, W'_{L-1}, \ldots, W'_1) \in \mathbb{R}^{d_\theta} \quad \text{and} \quad \Delta\theta = (\Delta W_L, \Delta W_{L-1}, \ldots, \Delta W_1) \in \mathbb{R}^{d_\theta}.$$

We use analogous notation for the product $W' = \mu(\theta')$, its displacement $\Delta W = W' - W$, the modified subsequence matrices $W'_{t \blacktriangleleft s} = W'_t W'_{t-1} \cdots W'_{s+1}$, and their displacements $\Delta W_{t \blacktriangleleft s} = W'_{t \blacktriangleleft s} - W_{t \blacktriangleleft s}$. All prebasis subspaces below are defined at θ. (They are different at θ'!)

Moves on the fiber replace a linear neural network with another that computes the same function. Here, we are interested in *basic one-matrix moves*, for which $\Delta\theta \in \phi_{lkjih}$ for some $\phi_{lkjih} \in \Theta_{\mathrm{O}}$. ("Basic" means "from the prebasis," not "simple.") Equivalently, a basic move replaces W_j with $W'_j = W_j + \Delta W_j$, where $\Delta W_j \in o_{lkjih}$ and $o_{lkjih} \in O_j$. Once we understand basic moves, other moves can be understood as sequences of basic moves. We are particularly interested in basic moves that change the rank list and move from one stratum to another.

For each subspace $o_{lkjih} \in O_j$, we ask: does a move with displacement $\Delta W_j \in o_{lkjih}$ change W? Which subsequence matrices change? Which subsequence matrices change rank? Which subsequence matrices undergo a change in rowspace or columnspace? The three lemmas below answer these questions, and Figure 6 summarizes the answers. (Figure 6 assumes the displacement ΔW_j is nonzero; the lemmas do not.) These answers explain why we chose subspaces of the form $o_{lkjih} = a_{lji} \otimes b_{k,j-1,h}$.

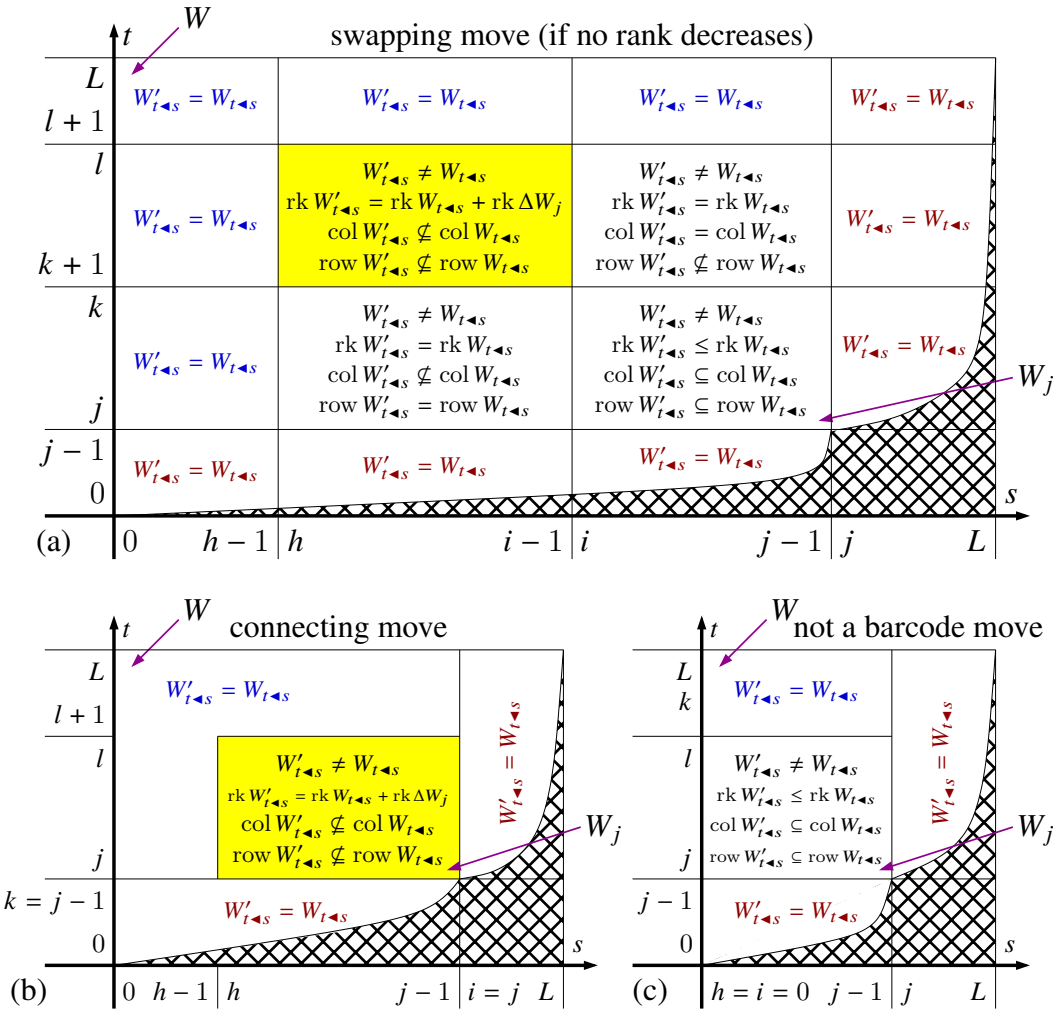


Figure 6 The influence of a basic move in which a factor matrix W_j undergoes a nonzero displacement $\Delta W_j \in o_{lkjih}$. The effects on the subsequence matrix $W_{t \blacktriangleleft s}$ are listed for every t and s with $L \geq t \geq s \geq 0$. These tables are triangular, though it's not obvious at first: the hatched regions represent unused zones where $t < s$. A yellow rectangle indicates which subsequence matrices increase in rank, constituting a barcode move if no rank decreases. The black font indicates where $W'_{t \blacktriangleleft s} \neq W_{t \blacktriangleleft s}$. The red font indicates where $W'_{t \blacktriangleleft s} = W_{t \blacktriangleleft s}$ because the matrix W_j is not a factor in $W_{t \blacktriangleleft s}$. The blue font indicates where $W'_{t \blacktriangleleft s} = W_{t \blacktriangleleft s}$ by properties of the flow relationships. (a) Table for the case where $L > l > k \geq j > i > h > 0$. An example of a swapping move. (b) The third row disappears if $k + 1 = j$, and the third column disappears if $i = j$. When $k + 1 = j = i$, the move is a connecting move. (c) The second row disappears if $k \geq l$, and the second column disappears if $i \leq h$. If either inequality holds, no subsequence matrix rank increases; the move is not a barcode move. The first column disappears if $h = 0$. (The first row disappears if $l = L$, though we don't depict that case here. If $h = 0$ and $l = L$, then $W' \neq W$ and we move off the fiber.)

▶ **Lemma 3.** *Given $L \geq t \geq s \geq 0$, $W'_{t \blacktriangleleft s} = W_{t \blacktriangleleft s}$ if and only if $\Delta W_j = 0$ or $j \notin [s+1, t]$ or $t > l$ or $s < h$. Moreover, if none of those four conditions holds, then $\operatorname{rk} \Delta W_{t \blacktriangleleft s} = \operatorname{rk} \Delta W_j$.*

▶ **Lemma 4.** *Given $L \geq t \geq s \geq 0$, $\operatorname{col} \Delta W_{t \blacktriangleleft s} \subseteq \operatorname{col} W_{t \blacktriangleleft s}$ (equivalently, $\operatorname{col} W'_{t \blacktriangleleft s} \subseteq \operatorname{col} W_{t \blacktriangleleft s}$) if and only if $\Delta W_j = 0$ or $j \notin [s+1, t]$ or $t > l$ or $s < h$ or $s \geq i$. Moreover, if none of those five conditions holds, then $\operatorname{col} \Delta W_{t \blacktriangleleft s} \cap \operatorname{col} W_{t \blacktriangleleft s} = \{\mathbf{0}\}$.*

Symmetrically, $\operatorname{row} \Delta W_{t \blacktriangleleft s} \subseteq \operatorname{row} W_{t \blacktriangleleft s}$ (equivalently, $\operatorname{row} W'_{t \blacktriangleleft s} \subseteq \operatorname{row} W_{t \blacktriangleleft s}$) if and only if $\Delta W_j = 0$ or $j \notin [s+1, t]$ or $t > l$ or $s < h$ or $t \leq k$. Moreover, if none of those five conditions holds, then $\operatorname{row} \Delta W_{t \blacktriangleleft s} \cap \operatorname{row} W_{t \blacktriangleleft s} = \{\mathbf{0}\}$.

▶ **Lemma 5.** *Given $L \geq t \geq s \geq 0$, if $t \in [j, k]$ and $s \in [i, j-1]$, then $\operatorname{rk} W'_{t \blacktriangleleft s} \leq \operatorname{rk} W_{t \blacktriangleleft s}$ and if $\|\Delta W_j\|$ is sufficiently small (in any matrix norm), $\operatorname{rk} W'_{t \blacktriangleleft s} = \operatorname{rk} W_{t \blacktriangleleft s}$.*

If $\Delta W_j \neq 0$ and $t \in [k+1, l]$ and $s \in [h, i-1]$, then $\operatorname{rk} W'_{t \blacktriangleleft s} = \operatorname{rk} W_{t \blacktriangleleft s} + \operatorname{rk} \Delta W_j$.

In all other cases, $\operatorname{rk} W'_{t \blacktriangleleft s} = \operatorname{rk} W_{t \blacktriangleleft s}$.

As $W = W_{L \blacktriangleleft 0}$, Lemma 3 tells us that a basic move moves off the fiber $\mu^{-1}(W)$ if and only if $\Delta\theta$ is a nonzero displacement from a subspace ϕ_{lkjih} with $l = L$ and $h = 0$. This motivates us to define a set of one-matrix subspaces that signify basic moves that stay on the fiber.

$$\Theta_{\mathrm{O}}^{\mathrm{fiber}} = \{\phi_{lkjih} \in \Theta_{\mathrm{O}} : L > l \text{ or } h > 0\}.$$

A *discrete move* is a move that changes the ranks of one or more subsequence matrices. If a discrete move stays on the fiber, it moves from one stratum to a different stratum. Lemma 5 tells us two ways a basic move can be discrete. Every subsequence matrix $W_{t \blacktriangleleft s}$ with $t \in [k+1, l]$ and $s \in [h, i-1]$ increases in rank, and their ranks all increase by the same amount: the rank of ΔW_j. A subsequence matrix $W_{t \blacktriangleleft s}$ with $t \in [j, k]$ and $s \in [i, j-1]$ *might* decrease in rank, but not if $\|\Delta W_j\|$ is sufficiently small.

A *small move* is a move that does not decrease the rank of any subsequence matrix. (Rather than fuss over "is $\|\Delta W_j\|$ small enough?", we define "small move" to meet our needs.) If a small, discrete move stays on the fiber, it moves to a stratum of higher dimension.

By Lemma 5, a small, basic move with displacement $\Delta\theta \in \phi_{lkjih}$ is discrete if and only if $\Delta\theta \neq \mathbf{0}$, $l > k$, and $i > h$ (so $[k+1, l] \neq \emptyset$ and $[h, i-1] \neq \emptyset$). We define a set of one-matrix subspaces that signify basic moves that increase the rank of some subsequence matrix.

$$\Theta_{\mathrm{O}}^{\mathrm{barc}} = \{\phi_{lkjih} \in \Theta_{\mathrm{O}} : l > k \text{ and } i > h\} = \{\phi_{lkjih} \neq \{\mathbf{0}\} : L \geq l \geq k+1 \geq j \geq i > h \geq 0\}.$$

6 Barcode moves: Walking the hierarchy of strata

To understand how the strata are connected to each other in the rank stratification of a fiber, it suffices to study small moves with displacements from subspaces in $\Theta_{\mathrm{O}}^{\mathrm{fiber}} \cap \Theta_{\mathrm{O}}^{\mathrm{barc}}$. Given a subspace $\phi_{lkjih} \in \Theta_{\mathrm{O}}^{\mathrm{fiber}} \cap \Theta_{\mathrm{O}}^{\mathrm{barc}}$, let $\theta + \phi_{lkjih}$ denote the *affine subspace* produced by translating ϕ_{lkjih} to pass through θ. By Lemma 3, $\theta + \phi_{lkjih} \subseteq \mu^{-1}(W)$, but by Lemma 5, no point in $\theta + \phi_{lkjih}$ except θ lies on the stratum S that contains θ. Whereas given a subspace $\phi_{lkjih} \in \Theta_{\mathrm{O}}^{\mathrm{fiber}} \setminus \Theta_{\mathrm{O}}^{\mathrm{barc}}$, $\theta + \phi_{lkjih} \subseteq \bar{S}$. These facts give us some hints about the connections among the strata and the shape of the stratification (albeit without addressing its curvature).

Interestingly, although a small, basic move may change the ranks of many subsequence matrices, at most four interval counts change. Recall the identity (1): $\omega_{ts} = \operatorname{rk} W_{t \blacktriangleleft s} - \operatorname{rk} W_{t \blacktriangleleft s-1} - \operatorname{rk} W_{t+1 \blacktriangleleft s} + \operatorname{rk} W_{t+1 \blacktriangleleft s-1}$. If all four ranks increase by $\operatorname{rk} \Delta W_j$, or exactly two ranks with opposite signs do, then ω_{ts} does not change.

For a small displacement $\Delta\theta \in \phi_{lkjih} \in \Theta_{\mathrm{O}}^{\mathrm{barc}}$, it is straightforward to check that ω_{kh} and ω_{li} decrease by $\operatorname{rk}\Delta W_j$, ω_{lh} and ω_{ki} increase by $\operatorname{rk}\Delta W_j$, and no other interval count changes. The barcode encodes the changes made by small, basic, discrete moves more elegantly than the rank list does. (The rank list is a *summed-area table* [6] of the interval counts.)

A *barcode move* is a small, basic, discrete move. There are two types of barcode moves: connecting moves and swapping moves. Every subspace $\phi_{lkjih} \in \Theta_{\mathrm{O}}$ has indices satisfying $k+1 \geq j \geq i$. A *connecting move* is a small move with a nonzero displacement $\Delta\theta \in \phi_{lkjih} \in \Theta_{\mathrm{O}}^{\mathrm{barc}}$ in the case where $k+1 = j = i$. In a connecting move, ω_{ki} does not exist (as $k < i$) and only three interval counts change. Figure 7 illustrates two examples of connecting moves and offers an interpretation in terms of a changing barcode: a connecting move deletes $\operatorname{rk}\Delta W_j$ copies of the interval $[h,k]$ and $\operatorname{rk}\Delta W_j$ copies of the interval $[i,l]$, and replaces them with $\operatorname{rk}\Delta W_j$ copies of the interval $[h,l]$. We think of this as connecting the intervals $[h,k]$ and $[i,l]$ together with an added link $[j-1,j] = [k,i]$ to create an interval $[h,l]$; hence the name "connecting move." (There is much intuition that can be gleaned from a careful study of the figure that is hard to explain in words.) The *rank* of a connecting move is $\operatorname{rk}\Delta W_j$.

A *swapping move* is a small move with a nonzero displacement $\Delta\theta \in \phi_{lkjih} \in \Theta_{\mathrm{O}}^{\mathrm{barc}}$ in the case where $k \geq i$. A swapping move changes four interval counts. Figure 8 illustrates two examples. A swapping move splices $\operatorname{rk}\Delta W_j$ copies of the interval $[h,k]$ with $\operatorname{rk}\Delta W_j$ copies of the interval $[i,l]$, replacing them with $\operatorname{rk}\Delta W_j$ copies of the interval $[h,l]$ (which is longer than both of the replaced intervals) and $\operatorname{rk}\Delta W_j$ copies of the interval $[i,k]$ (which is shorter than both). In effect, the interval endpoints are swapped. (Again, careful study of the figure is rewarding.) The *rank* of a swapping move is $\operatorname{rk}\Delta W_j$.

Each barcode move has an effect on the rank list (and the barcode) that depends solely on the indices h, i, k, and l and the rank of ΔW_j. This motivates the idea of an *abstract move* that maps one rank list to another rank list, and one barcode to another barcode, divorced entirely from any geometry. A *rank-c abstract barcode move* takes a valid rank list $\underline{r}$ and an index tuple (l,k,i,h) satisfying $L \geq l \geq k+1 \geq i > h \geq 0$, and yields the modified rank list $\underline{z}$ produced by setting $z_{t\blacktriangleleft s} = r_{t\blacktriangleleft s} + c$ for $t \in [k+1,l], s \in [h,i-1]$; for the other ranks, $z_{t\blacktriangleleft s} = r_{t\blacktriangleleft s}$. Equivalently, it takes the barcode for $\underline{r}$, decreases ω_{li} and ω_{kh} by c, and increases ω_{lh} and ω_{ki} by c – except that if $k+1 = i$, then ω_{ki} does not exist and only three interval counts change (an abstract connecting move). An abstract barcode move produces a valid rank list if and only if $\omega_{li} \geq c$ and $\omega_{kh} \geq c$ prior to the move.

We think the most interesting aspect of this paper is that the hierarchy of strata is captured by these simple discrete operations on barcodes – barcodes that indicate the flow of information through the neural network. Barcode moves create farther-reaching flows and may offer network training algorithms more opportunities to succeed.

7 Revealing the hierarchy of strata

We return to the hard part of Theorem 1, the claim that statement C implies statement D: if $\underline{r} \leq \underline{z}$, there exists a sequence of rank-one abstract barcode moves that takes us from $\underline{r}$ to $\underline{z}$. This claim is a bridge to showing that $S_{\underline{r}} \subseteq \bar{S}_{\underline{z}}$ if and only if $\underline{r} \leq \underline{z}$.

Given rank lists $\underline{r}$ and $\underline{z}$ with $\underline{r} < \underline{z}$, finding a sequence of abstract moves that takes us from $\underline{r}$ to $\underline{z}$ is a puzzle that took us three months to solve. The full proof does not fit here, but we include the constructive part of the proof: the algorithm FindLastMove in Figure 9 finds a rank list $\underline{u}$ such that $\underline{r} \leq \underline{u} < \underline{z}$ and a single rank-one abstract barcode move takes us from $\underline{u}$ to $\underline{z}$. Building on this step, a simple recursive algorithm, FindDAGPath in Figure 9, finds a sequence of abstract barcode moves that take us from $\underline{r}$ to $\underline{z}$ (computing the sequence in reverse order, from $\underline{z}$ to $\underline{r}$). The proof of correctness of FindDAGPath, and thus the proof that C implies D, follows by induction.

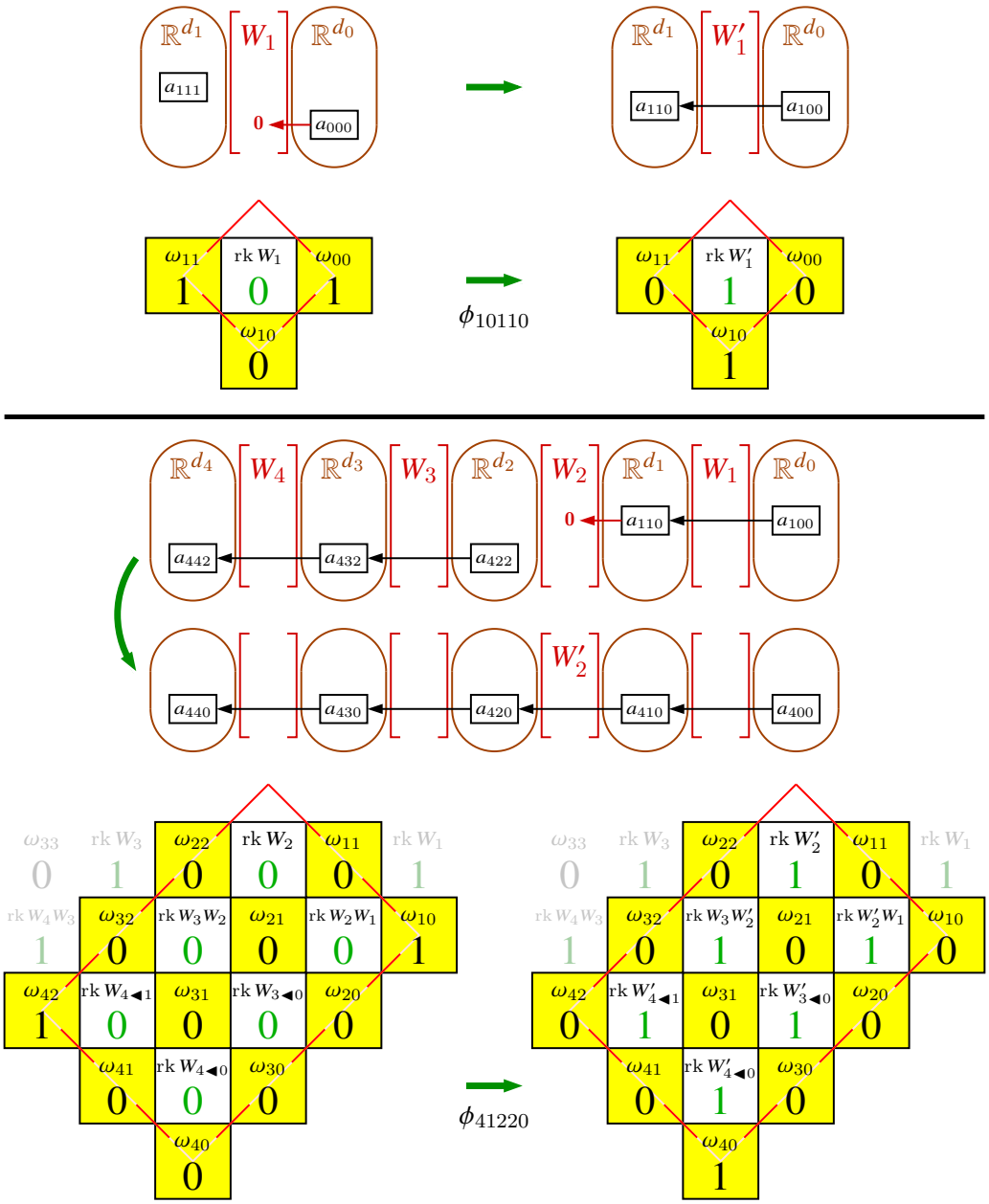


Figure 7 Two examples of connecting moves. The top example is the simplest example possible: W_1 has been perturbed to increase its rank from zero to one. In the bottom example, W_2 has been perturbed. In both examples, the perturbation of W_j replaces two intervals $[h, j-1]$ and $[j, l]$ with a single interval $[h, l]$. Three interval counts change, at three of the four corners of the red rectangle: ω_{lj} and $\omega_{j-1,h}$ decrease by one, and ω_{lh} increases by one. The ranks of the subsequence matrices $W_{t\blacktriangleleft s}$ increase by one for all $t \in [j, l]$ and $s \in [h, j-1]$ (the ranks inside the red rectangle, including $\operatorname{rk} W_j$). Outside the red rectangle, all interval counts and matrix ranks are unchanged.

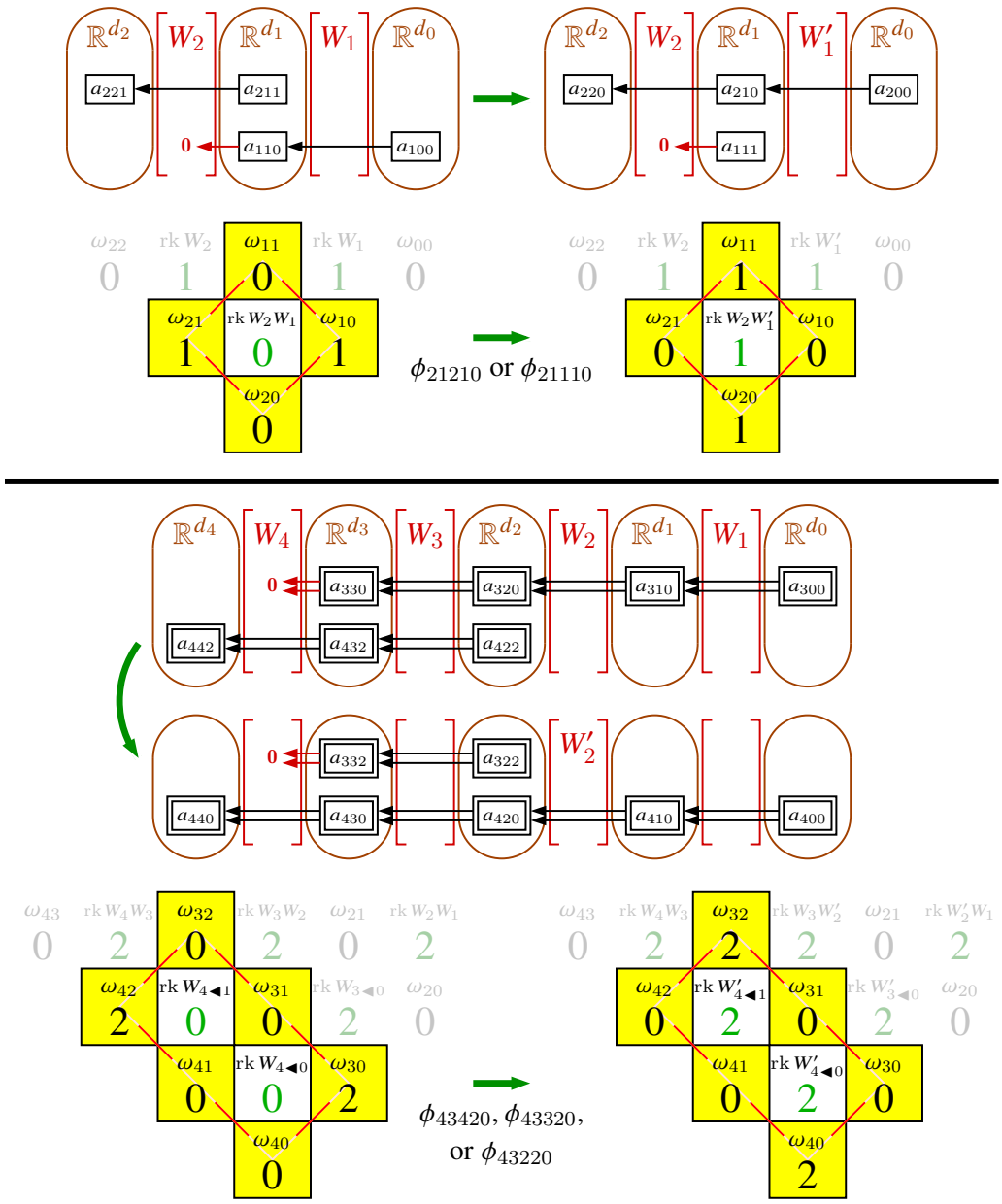


Figure 8 Two examples of swapping moves. In the top example – the simplest example possible – either W_1 or W_2 may be perturbed to cause the move. In the bottom example, any of W_2, W_3, or W_4 may be perturbed. The intervals $[h,k]$ and $[i,l]$ are replaced by an interval $[h,l]$, longer than both original intervals, and an interval $[i,k]$, shorter than both. Four interval counts change, at the four corners of the red rectangle: ω_{kh} and ω_{li} decrease, and ω_{lh} and ω_{ki} increase. The ranks of the subsequence matrices $W_{t\blacktriangleleft s}$ increase for all $t \in [k+1,l]$ and $s \in [h,i-1]$ (the ranks inside the red rectangle). Outside the red rectangle, all interval counts and matrix ranks are unchanged.

FindLastMove($\underline{r}, \underline{z}$)
// Given valid rank lists $\underline{r}$ and $\underline{z}$ such that $\underline{r} < \underline{z}$, returns a valid rank list $\underline{u}$ such that
// $\underline{r} \le \underline{u} < \underline{z}$ and a rank-one abstract barcode move takes us from $\underline{u}$ to $\underline{z}$.
for all t and s satisfying $L \ge t \ge s \ge 0$
    // From $\underline{r}$ and $\underline{z}$, compute all the interval counts with (1).
    $\omega^r_{ts} \leftarrow r_{t\blacktriangleleft s} - r_{t\blacktriangleleft s-1} - r_{t+1\blacktriangleleft s} + r_{t+1\blacktriangleleft s-1}$ // by convention, $r_{L+1\blacktriangleleft s} = r_{t\blacktriangleleft -1} = 0$
    $\omega^z_{ts} \leftarrow z_{t\blacktriangleleft s} - z_{t\blacktriangleleft s-1} - z_{t+1\blacktriangleleft s} + z_{t+1\blacktriangleleft s-1}$
    $\Delta\omega_{ts} \leftarrow \omega^z_{ts} - \omega^r_{ts}$
    $\Delta r_{t\blacktriangleleft s} \leftarrow z_{t\blacktriangleleft s} - r_{t\blacktriangleleft s}$ // every $\Delta r_{t\blacktriangleleft s}$ is nonnegative, as $\underline{r} < \underline{z}$
    // Initialize $\underline{u}$ to be the same as $\underline{z}$.
    $u_{t\blacktriangleleft s} \leftarrow z_{t\blacktriangleleft s}$
$[h, l] \leftarrow$ the longest interval (i.e., maximize $l - h$) such that $\Delta r_{l\blacktriangleleft h} > 0$
$i' \leftarrow$ the smallest index in $[h+1, l]$ such that $i' = l$ or $\Delta\omega_{l-1,i'} > 0$ or $\Delta r_{l\blacktriangleleft i'} = 0$
if $i' = l$ **or** $\Delta\omega_{l-1,i'} > 0$
    $i \leftarrow i'$; $k \leftarrow l - 1$
else // $\Delta r_{l\blacktriangleleft i'} = 0$
    $k \leftarrow$ the greatest index in $[i'-1, l-2]$ such that $k = i' - 1$ or $\Delta\omega_{ki} > 0$ for some $i \in [h+1, i']$
    $i \leftarrow$ the smallest index in $[h+1, i']$ such that $i = k+1$ or $\Delta\omega_{ki} > 0$
// Update $\underline{u}$ to perform a reverse rank-1 barcode move with index tuple (l, k, i, h).
for $t \leftarrow k+1$ **to** l
    for $s \leftarrow h$ **to** $i - 1$
        $u_{t\blacktriangleleft s} \leftarrow z_{t\blacktriangleleft s} - 1$
return $\underline{u}$

FindDAGPath($\underline{r}, \underline{z}$)
// Given rank lists $\underline{r}$ and $\underline{z}$ such that $\underline{r} \le \underline{z}$, finds and writes rank lists proceeding
// from $\underline{r}$ to $\underline{z}$ by a sequence of rank-one abstract barcode moves.
if $\underline{r} \ne \underline{z}$
    $\underline{u} \leftarrow$ FindLastMove($\underline{r}, \underline{z}$)
    FindDAGPath($\underline{r}, \underline{u}$)
Write $\underline{z}$

Figure 9 Algorithm to find a sequence of rank-1 abstract barcode moves that proceed from a rank list $\underline{r}$ to a rank list $\underline{z}$, assuming that $\underline{r} \le \underline{z}$. FindDAGPath operates recursively, repeatedly invoking FindLastMove to identify the last move in the sequence.

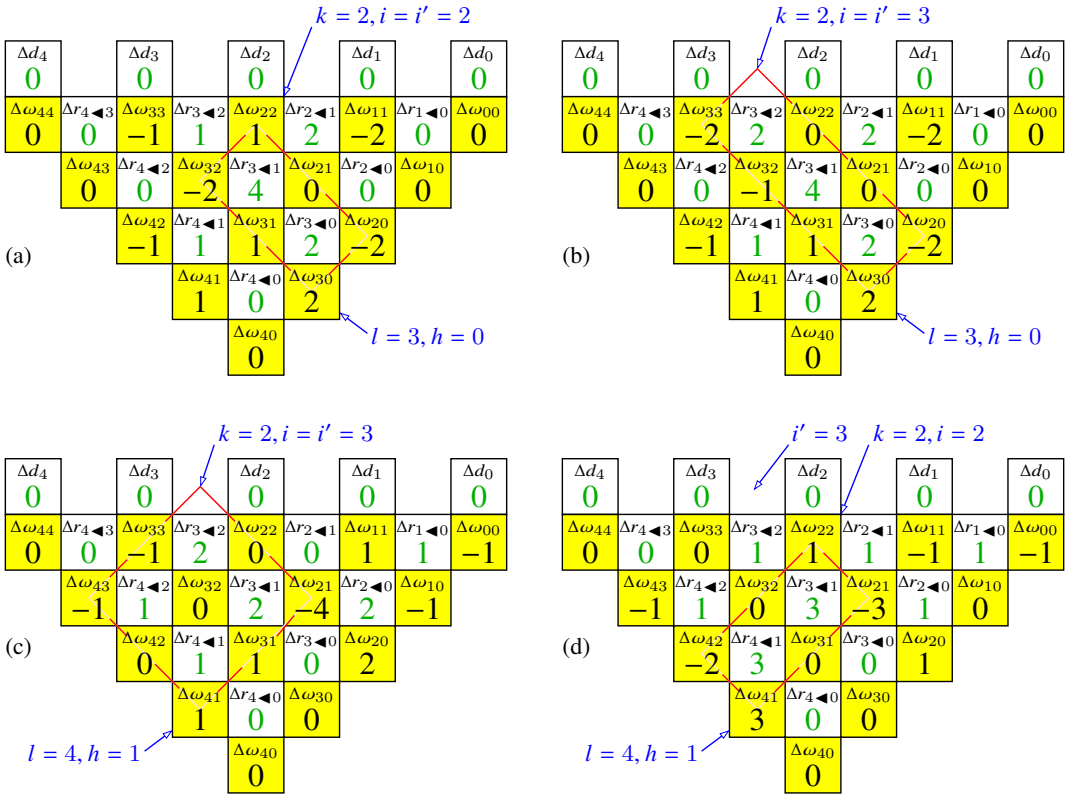


Figure 10 Four examples of reverse barcode moves found by the algorithm FindLastMove (shown prior to performing the reverse move). In each example, the algorithm finds a red rectangle such that its bottom corner lies at a bottommost positive $\Delta\omega_{lh}$, its top corner lies at a positive $\Delta\omega_{ki}$ or between two Δd's, and every Δr inside the rectangle is positive. In (b) and (c) the algorithm finds a reverse connecting move ($k = i - 1$), whereas in (a) and (d) it finds a reverse swapping move ($k \geq i$). In (a) and (b), Line 10 of FindLastMove determines the top corner of the rectangle (hence $k = l - 1$), whereas in (c) and (d), Lines 12 and 13 do (hence $k \leq l - 2$).

FindLastMove and our proofs use differences between the rank lists $\underline{r}$ and $\underline{z}$. Let the interval counts associated with $\underline{r}$ be ω^r_{ts} and the interval counts associated with $\underline{z}$ be ω^z_{ts}. Let

$$\Delta r_{t \blacktriangleleft s} = z_{t \blacktriangleleft s} - r_{t \blacktriangleleft s} \quad \text{and} \quad \Delta\omega_{ts} = \omega^z_{ts} - \omega^r_{ts} \quad \text{for all } t \text{ and } s \text{ satisfying } L \geq t \geq s \geq 0.$$

As we assume that $\underline{r} \leq \underline{z}$, no $\Delta r_{t \blacktriangleleft s}$ is negative. (No such constraint applies to $\Delta\omega_{ts}$.)

Figure 10 depicts examples of reverse moves found by FindLastMove, and thereby sheds light on Lines 7–13, the core of the algorithm. In each example, a red rectangle shows which reverse move the algorithm finds – specifically, the index tuple (l, k, i, h) of the move. Line 7 selects l and h so that the bottom corner of the red rectangle, $\Delta\omega_{lh}$, is a bottommost positive $\Delta\omega$. (Equivalently, the bottommost rank inside the rectangle, $\Delta r_{l \blacktriangleleft h}$, is a bottommost positive Δr.) Lines 8–13 search for values of k and i such that every Δr inside the rectangle is positive and the rectangle's top corner is either at a positive $\Delta\omega_{ki}$ or between two Δd's ($k = i - 1$) – these constraints guarantee that a reverse move yields a valid rank list. The hard part of our proof is showing that there always exists a rectangle satisfying these constraints.

Lines 8–9 check whether such a rectangle exists with $k = l - 1$. If so, Line 10 sets i to indicate the smallest such rectangle; see Figures 10(a) and (b) for examples. If not, the obstacle is some $\Delta r_{l \blacktriangleleft i'}$ equal to zero. In that case, Lines 12–13 find values of k and i that

satisfy the constraints; see Figures 10(c) and (d) for examples. Showing that Lines 12–13 always succeed is the part of the proof we must omit here. (It takes about two pages.)

We conclude with the hard part of Theorem 1.

▶ **Lemma 6.** *Let $\underline{r}$ and $\underline{z}$ be two valid rank lists for the same linear neural network (i.e., $r_{j\blacktriangleleft j} = z_{j\blacktriangleleft j} = d_j$ for all $j \in [0, L]$) such that $\underline{r} \leq \underline{z}$. Then there exists a sequence of valid rank lists that starts with $\underline{r}$ and ends with $\underline{z}$ such that each rank list after $\underline{r}$ can be obtained from the previous rank list in the sequence by a single rank-1 abstract barcode move. Moreover, the algorithm* FindDAGPath *in Figure 9 finds such a sequence.*

8 Final thoughts

Small moves are our tool for understanding stratum connections. But we speculate that longer moves on the fiber might be useful in practice to replace a neural network with an equivalent network more amenable to training by gradient descent algorithms, because barcode moves create farther-reaching flows, and longer moves make those flows stronger. A related idea is to modify gradient descent so it tends to move away from "bad" strata.

Given a smooth function $\ell(W)$, Trager, Kohn, and Bruna [23] call θ a *spurious critical point* if θ is a critical point of $\ell(\mu(\theta))$ but $\mu(\theta)$ is not a critical point of ℓ (where the domain of ℓ is restricted to matrices of rank $\min_{j=0}^{L} d_j$ or less). They show that there is no spurious critical point on $\mu^{-1}(W)$ if $\operatorname{rk} W = \min_{j=0}^{L} d_j$; also that there is no spurious critical point on any stratum $S_{\underline{r}}$ that satisfies, for some $j \in [1, L]$, both $r_{L\blacktriangleleft j} = d_L$ and $r_{j-1\blacktriangleleft 0} = d_0$.

We anticipate that geometric interpretations of cost functions in neural networks have the potential to become a thriving area, especially if we can incorporate ReLU activations.

References

1 Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 244–253, Stockholm, Sweden, July 2018. URL: https://proceedings.mlr.press/v80/arora18a.html.

2 Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit Regularization in Deep Matrix Factorization. In *Advances in Neural Information Processing Systems*, volume 32, pages 7411–7422, 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/c0c783b5fc0d7d808f1d14a6e9c8280d-Abstract.html.

3 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The Structure and Stability of Persistence Modules*. Springer, 2016. doi:10.1007/978-3-319-42545-0.

4 Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The Loss Surfaces of Multilayer Networks. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 192–204, San Diego, California, USA, May 2015. URL: https://proceedings.mlr.press/v38/choromanska15.html.

5 Hung-Hsu Chou, Carsten Gieshoff, Johannes Maly, and Holger Rauhut. Gradient Descent for Deep Matrix Factorization: Dynamics and Implicit Bias towards Low Rank. *Applied and Computational Harmonic Analysis*, 68:101595, 2024. doi:10.1016/j.acha.2023.101595.

6 Franklin C. Crow. Summed-Area Tables for Texture Mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '84)*, pages 207–212, Minneapolis, Minnesota, July 1984. doi:10.1145/800031.808600.

7 Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010. URL: http://www.ams.org/bookstore-getitem/item=MBK-69.

8 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological Persistence and Simplification. *Discrete & Computational Geometry*, 28(4):511–533, November 2002. doi:10.1007/s00454-002-2885-2.

9 Robert Ghrist. Barcodes: The Persistent Topology of Data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008. doi:10.1090/S0273-0979-07-01191-3.

10 Suriya Gunasekar, Blake E. Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. Implicit Regularization in Matrix Factorization. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6152–6160, 2017. doi:10.1109/ITA.2018.8503198.

11 Moritz Hardt and Tengyu Ma. Identity Matters in Deep Learning, July 2018. arXiv:1611.04231.

12 Ziwei Ji and Matus Telgarsky. Gradient Descent Aligns the Layers of Deep Linear Networks. In *7th International Conference on Learning Representations*, New Orleans, Louisiana, May 2019. URL: https://openreview.net/forum?id=HJflg30qKX.

13 Kenji Kawaguchi. Deep Learning without Poor Local Minima. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, volume 29, pages 586–594, 2016. URL: https://proceedings.neurips.cc/paper/2016/hash/f2fc990265c712c49d51a18a32b39f0c-Abstract.html.

14 Qianyi Li and Haim Sompolinsky. Statistical Mechanics of Deep Linear Neural Networks: The Backpropagating Kernel Renormalization. *Physical Review X*, 11:031059, September 2021. doi:10.1103/PhysRevX.11.031059.

15 Clara Löh. A Comment on the Structure of Graded Modules over Graded Principal Ideal Domains in the Context of Persistent Homology, February 2023. arXiv:2301.11756.

16 Stanisław Łojasiewicz. Ensembles Semi-Analytiques. Unpublished manuscript. A version typeset by Michel Coste is available at https://perso.univ-rennes1.fr/michel.coste/Lojasiewicz.pdf, July 1970.

17 Haihao Lu and Kenji Kawaguchi. Depth Creates No Bad Local Minima, May 2017. arXiv:1702.08580.

18 John Mather. Notes on Topological Stability. Lecture Notes. Reprinted in *Bulletin of the American Mathematical Society* 49(4):475–506, October 2012, 1970. doi:10.1090/S0273-0979-2012-01383-6.

19 Adityanarayanan Radhakrishnan, Eshaan Nichani, Daniel Irving Bernstein, and Caroline Uhler. On Alignment in Deep Linear Neural Networks, June 2020. arXiv:2003.06340v2.

20 Jonathan Richard Shewchuk and Sagnik Bhattacharya. The Geometry of the Set of Equivalent Linear Neural Networks, April 2024. doi:10.48550/arXiv.2404.14855.

21 Gilbert Strang. The Fundamental Theorem of Linear Algebra. *The American Mathematical Monthly*, 100(9):848–855, November 1993. doi:10.1080/00029890.1993.11990500.

22 René Thom. La Stabilité Topologique des Applications Polynomiales. *L'Enseignement Mathématique*, 8(1–2):24–33, 1962.

23 Matthew Trager, Kathlén Kohn, and Joan Bruna. Pure and Spurious Critical Points: A Geometric Study of Linear Networks. In *Eighth International Conference on Learning Representations*, Addis Ababa, Ethiopia, April 2020. URL: https://openreview.net/forum?id=rkgOlCVYvB.

24 Hassler Whitney. Elementary Structure of Real Analytic Varieties. *Annals of Mathematics*, 66(2):545–556, 1957. doi:10.2307/1969908.

25 Hassler Whitney. Local Properties of Analytic Varieties. In Stewart Scott Cairns, editor, *Differential and Combinatorial Topology*, pages 205–244. Princeton University Press, 1965. doi:10.1515/9781400874842-014.

26 Hassler Whitney. Tangents to an Analytic Variety. *Annals of Mathematics*, 81(2):496–549, 1965. doi:10.2307/1970400.

27 Li Zhang. Depth Creates No More Spurious Local Minima, January 2020. arXiv:1901.09827.

28 Afra Zomorodian and Gunnar E. Carlsson. Computing Persistent Homology. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 347–356, Brooklyn, New York, June 2004. doi:10.1145/997817.997870.

29 Afra Zomorodian and Gunnar E. Carlsson. Computing Persistent Homology. *Discrete & Computational Geometry*, 33(2):249–274, February 2005. doi:10.1007/s00454-004-1146-y.

Estimation of Conformal Metrics

Jérôme Taupin
Université Paris-Saclay, France
INRIA Saclay, Palaiseau, France

Abstract

We study deformations of the geodesic distances on a domain of $\mathbb{R}^N$ induced by a function called conformal factor. We show that under a positive reach assumption on the domain (not necessarily a submanifold) and mild assumptions on the conformal factor, geodesics for the conformal metric have good regularity properties in the form of a lower bounded reach. This regularity allows for efficient estimation of the conformal metric from a random point cloud with a relative error proportional to the Hausdorff distance between the point cloud and the original domain. We then establish convergence rates of order $n^{-1/d}$ that are close to sharp when the intrinsic dimension d of the domain is large, for an estimator that can be computed in $\mathcal{O}(n^2)$ time. Finally, this paper includes a useful equivalence result between ball graphs and nearest-neighbors graphs when assuming Ahlfors regularity of the sampling measure, allowing to transpose results from one setting to another.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geometric inference, metric estimation, conformal metric, geodesics, sets of positive reach

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.92

Related Version *Full Version*: `https://hal.science/hal-05516466v1` [17]

1 Introduction

This paper studies metrics over subsets of the Euclidean space $(\mathbb{R}^N, \|\cdot\|)$ obtained by conformal deformation of the shortest-path metric via a positive function. We are particularly interested in the regularity of such metrics and in their estimation via i.i.d. sampling of points.

▶ **Definition 1.1.** *Let $M \subset \mathbb{R}^N$ be a closed path-connected* domain *and $f : M \to \mathbb{R}_+^*$ be a* conformal factor. *For all $x, y \in M$ the* conformal distance *between x and y over M via f is defined as*

$$D_{M,f}(x,y) \overset{def}{=} \inf_{\gamma \in \Gamma_M(x,y)} \int_I f(\gamma(t)) \|\dot{\gamma}(t)\| \mathrm{d}t \tag{1}$$

where $\Gamma_M(x,y)$ is the set of all Lipschitz paths $\gamma : I \to M$ where $I = [a,b]$ is a nontrivial segment, $\gamma(a) = x$ and $\gamma(b) = y$. The quantity minimized over all paths γ is denoted

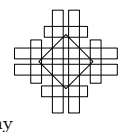
$$|\gamma|_f \overset{def}{=} \int_I f(\gamma(t)) \|\dot{\gamma}(t)\| \mathrm{d}t$$

and referred to as the conformal length of the curve γ via f. In the case where $f = 1$, we write $D_M(x,y) = D_{M,1}(x,y)$ and $|\gamma| = \int_I \|\dot{\gamma}\|$, respectively the distance between x and y induced by the ambient metric over M and the Euclidean length of a curve γ.

In the following, to ensure regularity of the conformal metric, the domain M is assumed to have positive reach τ_M, which we recall is defined as the supremum of all $r > 0$ such that any point in the offset $M^r = \{x \in \mathbb{R}^N : d(x,M) < r\}$ has a unique projection onto M – where $d(x,M) = \inf_{y \in M} \|x - y\|$ denotes the distance from point x to subset M – see [10]. Moreover, the conformal factor f is assumed to be κ-Lipschitz and lower bounded by $f_{\min} > 0$. These are the sole assumptions used in this paper regarding M and f.



© Jérôme Taupin;
licensed under Creative Commons License CC-BY 4.0





Notice that f is defined over M in Definition 1.1 as paths are constrained to the domain. Since any Lipschitz function defined over a subset of $\mathbb{R}^N$ can be extended to the whole space preserving its Lipschitz property and lower bound (see [16, Theorem 1] for instance), one may assume without loss of generality that f is defined over the entire space $\mathbb{R}^N$, which is useful to estimate the conformal metric efficiently. The connectedness and positive reach of M ensure that for any endpoints $x, y \in M$, there exists a Lipschitz path from x to y in M, so that $\Gamma_M(x, y)$ is always nonempty and $D_{M,f}(x, y)$ is always finite. This can be deduced from Lemma 2.1 below. Most of the time, a path γ is chosen to be parameterized with constant velocity, i.e., $\|\dot{\gamma}\|$ is constant over I, with I being either $[0, 1]$ or $[0, |\gamma|]$. The latter is referred to as an *arc-length* parameterization, where $\|\dot{\gamma}\| = 1$ almost everywhere.

The *induced metric* $D_M = D_{M,1}$ is the metric induced by the ambient space $\mathbb{R}^N$ onto M. If M is a $\mathcal{C}^1$ submanifold of $\mathbb{R}^N$ and f is $\mathcal{C}^1$, $D_{M,f}$ is a Riemannian metric with tensor $f(x)^2 \cdot g_x$ at point $x \in M$ where g_x is the tensor of the induced metric D_M at x. The term "conformal" used in this paper is borrowed from the Riemannian literature. Finally, if μ is a measure over a submanifold M with density ρ with regard to (w.r.t.) the volume form of M and f is a negative power of the density ρ, $D_{M,f}$ has already been studied and is sometimes referred to as the *Fermat distance* [13]. This particular case is one of the main motivations for this article as the Fermat distance has been used for various practical applications, see for instance [12] and the references therein. Particular examples of the conformal factor f are discussed in Section 5.

Conformal metrics are included in the more general class of length-spaces, for which it is known that the infimum in Definition 1.1 is in fact a minimum [7, Theorem 2.5.23]. That is, for all $x, y \in M$ there exists a path $\gamma \in \Gamma_M(x, y)$ such that $D_{M,f}(x, y) = |\gamma|_f$, called a *minimizing geodesic* – and shortened to geodesic in this paper for brevity. Denote $\Gamma^\star_{M,f}(x, y)$ the set of such geodesics between two endpoints x and y along with $\Gamma^\star_{M,f} = \bigcup_{x \neq y \in M} \Gamma^\star_{M,f}(x, y)$ the set of all geodesics w.r.t. the conformal metric. We discuss in Section 2 the regularity of geodesics w.r.t. $D_{M,f}$ and show in Proposition 2.4 that under the aforementioned assumptions on M and f, geodesics have positive reach that is lower bounded by an explicit constant depending on the reach of the domain and on the conformal factor. In particular, any geodesic may be parameterized as a $\mathcal{C}^{1,1}$ curve with an explicit upper bound on the Lipschitz constant of its first derivative.

We then show in Section 3 that the conformal metric can be approached using polygonal paths on a weighted graph built from a point cloud $X \subset M$, provided that the graph only contains edges of length at most some threshold r and that X is close to M in Hausdorff distance. The small edges condition ensures that paths on the graph cannot venture too far outside the domain. This kind of construction is the same as the one used for the Isomap algorithm [5], although we allow more generality by adapting the weights to the conformal factor f. When $f = 1$ and M is a geodesically convex $\mathcal{C}^2$ submanifold, [5] provides a relative error bound of order $\frac{r^2}{\tau^2} + \frac{\rho}{r}$ where τ is the minimum radius of curvature of M and ρ denotes the Hausdorff distance between M and X. The term depending on ρ can be made quadratic as shown in [2], where an assumption called geodesic smoothness that is slightly weaker than that of a positive reach allows to obtain a bound of order $Ar + \frac{\rho^2}{r^2}$ where A depends on the assumption. This assumption however lacks precision when comparing distances at a local scale which results in the first term being of order r instead of r^2. By using a positive reach assumption instead, the upper bound can be further improved to $\frac{r^2}{\tau^2} + \frac{\rho^2}{r^2}$ according to [3] where τ is the reach of the domain, although the setup also implicitly assumes that the domain M is a smooth manifold isometric to a convex domain. Going back to our setup, we adapt these prior works to fit any conformal change. Under some condition on the

weight function used, that in particular always holds for the induced metric, we establish in Theorem 3.5 the same upper bound on the approximation error as in [3], that is $\frac{r^2}{\tau^2} + \frac{\rho^2}{r^2}$ where τ is the explicit lower bound on the reach of any geodesic provided by Proposition 2.4. This result emphasizes the fact that the only assumption on the domain needed for this level of precision is that of a positive reach. The approximation error can thus be made proportional to the Hausdorff distance between the point cloud and the domain by choosing appropriately $r \asymp \sqrt{\tau\ \rho}$. When the conformal factor f is unknown, replacing it with an estimate g does not alter the results provided that g is close enough to f, as shown in Lemma 3.6.

Assuming that $X = X_n$ is the outcome of $n \geq 1$ i.i.d. samples from a d-standard measure μ over M, we study in Section 4 the estimator of $D_{M,f}$ built on X_n following the construction of Section 3. This estimator is shown in Theorem 4.2 to converge to $D_{M,f}$ at a rate of $n^{-1/d}$ provided that f is known or can be estimated with same rate. This rate follows from Theorem 3.5 and the fact that X_n is known to converge to M in Hausdorff distance at a rate of $n^{-1/d}$. In particular, this allows for efficient estimation of the induced metric with the sole assumption that M has positive reach. This convergence is shown using ball graphs. However, practical estimation is made easier by using nearest-neighbors graphs instead, in part because it does not require to know the intrinsic dimension d to obtain an optimal convergence speed. For this reason, we show in Theorem 4.5 that it is possible to retrieve the same convergence speed when replacing the ball graph in the estimator with a k-nearest-neighbors graph with $k \asymp \sqrt{n}$. To do so, we establish an equivalence with high probability between nearest-neighbors graphs and ball graphs when their respective parameters are properly scaled and the underlying measure is d-Ahlfors, see Proposition 4.4. Finally, the nearest-neighbors estimator of the conformal distance between two fixed points may be computed in $\mathcal{O}(n^2)$ if the time complexity of evaluating f is considered constant. Under the stronger assumption that M is a $\mathcal{C}^k$ submanifold of dimension d with $k \geq 2$, the minimax optimal convergence rate for the induced metric is known to be of order $n^{-k/d}$ [1]. When only assuming positive reach and no differential structure on M, the discussion of the optimal minimax rate proves to be harder as we are not able to match the upper bound of $n^{-1/d}$ and obtain a lower bound of order $n^{-1/(d-1/2)}$ instead, see Theorem 4.6. We discuss with more details the comparison between our setup and the $\mathcal{C}^k$ case in Section 4.4.

Technical proofs are deferred to the appendix which is available in the full version [17].

2 Regularity of the geodesics for the conformal metric

Recall that the term "geodesic" refers in this paper to a curve that achieves a global minimum of the conformal length. In this section we study the regularity of the geodesics w.r.t. $D_{M,f}$. We show that geodesics are $\mathcal{C}^{1,1}$ curve with reach bounded from below by a constant depending only on τ_M, κ and $f_{\min}$. This regularity property is crucial to obtain a good approximation of the geodesics by polygonal paths. The reach of a set can be characterized through the local distortion of the induced metric compared to the Euclidean metric.

▶ **Lemma 2.1** ([6, Theorem 1]). *The reach of M may be expressed as*

$$\tau_M = \sup\left\{ r > 0 \ : \ \forall x, y \in M,\ \|x-y\| < 2r \Rightarrow D_M(x,y) \leq 2r \arcsin\left(\frac{\|x-y\|}{2r}\right)\right\} .$$

Given a path $\gamma \in \Gamma_M$ parameterized over an interval I and without self-intersection, denote τ_γ the reach of the curve $\gamma(I) \subset \mathbb{R}^N$ along with $D_\gamma = D_{\gamma(I)}$ the induced metric over the curve for short. $D_\gamma(x,y)$ is nothing more than the length of the curve $\gamma(I)$ between two intermediate point x and y. In particular if x and y are the endpoints of γ then $D_\gamma(x,y) = |\gamma|$.

One consequence of Lemma 2.1 is that the length of a geodesic between two points at most $2\tau_M$ apart is upper bounded by the length of an arc of radius τ_M between both points. That is, if $x, y \in M$ and $\|x - y\| < 2\tau_M$ then

$$D_M(x, y) \leq 2\tau_M \arcsin\left(\frac{\|x - y\|}{2\tau_M}\right) . \tag{2}$$

Moreover, if γ is a geodesic, it then induces a geodesic between any of the points it goes through, which implies that D_γ is exactly the restriction of D_M to the curve γ. Then, another consequence of the characterization of Lemma 2.1 is that the reach of M is the minimal reach of any geodesic, that is

$$\tau_M = \inf_{\gamma \in \Gamma^\star_M} \tau_\gamma .$$

We now introduce a notion of reach associated with the conformal deformation of M by f using the same point of view of the metric and its geodesics.

▶ **Definition 2.2.** *The* conformal reach *of M via f is defined as the minimal reach of any geodesic w.r.t. the conformal metric, that is*

$$\tau_{M,f} \overset{def}{=} \inf_{\gamma \in \Gamma^\star_{M,f}} \tau_\gamma .$$

In the case of the induced metric this notion coincides with the usual notion of reach, that is $\tau_{M,1} = \tau_M$. The characterization given by Lemma 2.1 also holds for the conformal reach.

▶ **Proposition 2.3.** *The conformal reach of M via f may be expressed as*

$$\tau_{M,f} = \sup \left\{ \begin{array}{c} r > 0 \ : \ \forall x, y \in M, \forall \gamma \in \Gamma^\star_{M,f}(x, y), \\ \|x - y\| < 2r \Rightarrow |\gamma| \leq 2r \arcsin\left(\dfrac{\|x - y\|}{2r}\right) \end{array} \right\} . \tag{3}$$

Beware that Equation (3) involves geodesic w.r.t. the conformal metric $D_{M,f}$, but compares their Euclidean length – not conformal – to the one of an arc of radius r. The proof is straightforward and included in the full version [17]. Using Proposition 2.3, we are able to lower bound the reach of any geodesic w.r.t. $D_{M,f}$.

▶ **Proposition 2.4.** *Assume that M has positive reach $\tau_M > 0$ and that f is κ-Lipschitz and lower bounded by $f_{\min} > 0$. Then any conformal geodesic γ w.r.t. $D_{M,f}$ has positive reach $\tau_\gamma > 0$. Precisely, the conformal reach of M via f is lower bounded as follows.*

$$\tau_{M,f} \geq \mathcal{T}_{M,f} \quad \textit{where} \quad \mathcal{T}_{M,f} \overset{def}{=} \min\left(\frac{\tau_M}{2} \ , \ \frac{f_{\min}}{8\kappa}\right) .$$

The reasoning behind Proposition 2.4 is the following. If the reach of a geodesic γ w.r.t. $D_{M,f}$ is small compared to τ_M, then according to Proposition 2.3 there exists a path in M significantly shorter than γ Euclidean-wise. If τ_γ is also small compared to $f_{\min}/\kappa$, then this path is shown to also have a shorter conformal length than γ due to the properties of f, which implies a contradiction with the geodesic nature of γ. This reasoning is detailed in [17, Appendix B.2]. Now, consider an arc-length parameterized geodesic $\gamma : I \to M$ w.r.t. $D_{M,f}$, i.e., such that $\|\dot\gamma\| = 1$ almost everywhere over I. Being a geodesic, γ has no self-intersection. Then, stating that γ has positive reach is equivalent to stating that γ is a $\mathcal{C}^{1,1}$ curve, i.e., that $\dot\gamma$ is Lipschitz w.r.t. the angular distance. Precisely, for all $t, s \in I$,

$$\angle(\dot\gamma(t), \dot\gamma(s)) \leq \frac{|t - s|}{\tau_\gamma} . \tag{4}$$

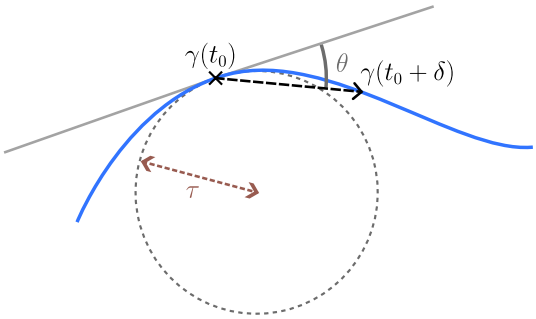


Figure 1 Bounding the angular velocity of a path with positive reach.

See [10, Remark 4.20] and [15, Theorem 4] for references. Equation (4) allows to obtain a finer control on the approximation error of the polygonal paths on the graph by upper bounding efficiently the difference between small successive steps $\gamma(t) - \gamma(t-\delta)$ and $\gamma(t+\delta) - \gamma(t)$ of a geodesic path.

▶ **Lemma 2.5.** *Let $\gamma : [0, |\gamma|] \to \mathbb{R}^N$ be an arc-length parameterized curve without self-intersection and with positive reach τ_γ. Then for all $t_0 \in [0, |\gamma|]$ and $\delta \in (0, \frac{\pi}{2}\tau_\gamma]$ such that $[t_0 - \delta, t_0 + \delta] \subset [0, |\gamma|]$, the angle between the velocity vector of γ at t_0 and the direction from $\gamma(t_0)$ to $\gamma(t_0 + \delta)$ is upper bounded as follows:*

$$\angle\big(\gamma(t_0+\delta) - \gamma(t_0), \dot\gamma(t_0)\big) \leq \frac{\delta}{2\tau_\gamma} \ . \tag{5}$$

Moreover, denoting $v^- = \gamma(t_0) - \gamma(t_0 - \delta)$ and $v^+ = \gamma(t_0 + \delta) - \gamma(t_0)$, the difference between small steps of the path on both side from t_0 is upper bounded as follows:

$$\left\| \frac{v^+}{\|v^+\|} - \frac{v^-}{\|v^-\|} \right\| \leq \frac{1}{\tau_\gamma} \min\left(\|v^+\|, \|v^-\|\right) \ . \tag{6}$$

Lemma 2.5 is key to obtain an approximation error of the conformal metric proportional to the Hausdorff distance between the domain and a point cloud. A proof is provided in [17, Appendix A.1] and Equation (5) is illustrated by Figure 1.

3 Polygonal approximation of the conformal metric

Consider a point cloud $X \subset M$ of points sampled from the domain. If the point cloud approximates well the domain, it is possible to estimate the conformal distance over M through weighted polygonal paths built over X within a small margin of error. Assuming that f is known, the edges of the polygonal path are weighted using f to approximate the conformal distance between their endpoints. The degree of approximation of M by X is measured using the Hausdorff distance

$$d_{\mathrm{H}}(M, X) \stackrel{\text{def}}{=} \max\left(\sup_{x\in M} d(x, X) \ , \ \sup_{x\in X} d(x, M)\right)$$

which simplifies to $d_{\mathrm{H}}(M, X) = \sup_{x\in M} d(x, X)$ assuming that $X \subset M$. By definition, any point in M is at distance at most $d_{\mathrm{H}}(M, X)$ from a point in X.

The idea behind the polygonal approximation is the following. Consider a geodesic $\gamma \in \Gamma^\star_{M,f}(x, y)$ between two points $x, y \in M$. If $d_{\mathrm{H}}(M, X)$ is small, there exists a polygonal path over X that follows closely the trajectory of γ. Assuming that the edges of this path

are equipped with appropriate weights to simulate the conformal distance between their endpoints, the total weight of the polygonal path should be close to the conformal length $|\gamma|_f$. Moreover, since γ is a geodesic, polygonal paths should not be able to have a weight much smaller than $|\gamma|_f$. This is ensured by using only short edges, which allows the weights to approximate well the conformal length by preventing shortcuts outside the domain. In the end, the shortest weighted path on the graph is expected to retrieve $|\gamma|_f = D_{M,f}(x, y)$.

3.1 Weighted graphs

Let us now describe formally this construction. The polygonal approximation of $D_{M,f}$ is defined as the metric of an appropriate weighted graph on a point cloud X.

▶ **Definition 3.1.** *Let $X \subset \mathbb{R}^N$ be a finite point cloud.*

- *For all $r > 0$, the* r-ball graph *over X is denoted $G_r(X)$ and is the graph of vertex set X and with an edge between points x and y if and only if $\|x - y\| \leq r$.*
- *For all integer $k \geq 1$, the* k-nearest-neighbors graph *(or k-NN graph) over X is denoted $\mathcal{G}_k(X)$ and is the graph of vertex set X with an edge between two points x and y if and only if x (or y) is among the k nearest points of X to y (or x) excluding self.*

In both cases, the parameter r or k is referred to as the threshold *of the graph.*

Ball graphs are more convenient to study whereas NN graphs are more practical, see Section 4.2. We define the polygonal metric in both contexts. The graph used is equipped with a weight function to approximate the conformal length of its edges.

▶ **Definition 3.2.** *Given a function $f : \mathbb{R}^N \to \mathbb{R}_+^*$, consider the weight functions*

$$w_{f,q}(x, y) \overset{def}{=} \frac{\|x - y\|}{2(q-1)} \left(f(x) + 2 \sum_{k=2}^{q-1} f\left(\frac{q-k}{q-1} x + \frac{k-1}{q-1} y \right) + f(y) \right)$$

defined for any $q \geq 2$ and $x, y \in \mathbb{R}^N$, along with

$$w_{f,\infty}(x, y) \overset{def}{=} \|x - y\| \int_0^1 f\big((1-t)x + ty\big) \mathrm{d}t \ .$$

The parameter $q \in \{2, \ldots, \infty\}$ is referred to as the resolution *of the weights.*

The weight function $w_{f,q}$ is meant to approximate the conformal distance between the endpoints of an edge. Proposition 3.3 below shows that it is indeed the case when the endpoints are close, which is the reason why graphs with small edges are used. Recall that we have assumed without loss of generality that f is defined over the whole space, allowing its evaluation outside M when $q \neq 2$. As q grows, the weight $w_{f,q}$ becomes more accurate and converges to the weight with infinite resolution $w_{f,\infty}$, the latter being exactly the conformal length of the straight path. However, the computational cost is linear with q and there is no closed-form formula for $w_{f,\infty}$ in general. On the other hand,

$$w_{f,2}(x, y) = \|x - y\| \frac{f(x) + f(y)}{2}$$

is the simplest choice for the weights and may be more suitable if evaluating f is possible only at points in X. Note that the weight $w_{f,q}$ is the optimal approximation of $w_{f,\infty}$ using only q samples of the Lipschitz function f. Under stronger assumptions on f such as $\mathcal{C}^k$ regularity, other definitions would be better suited.

▶ **Proposition 3.3.** *For all $x, y \in M$ such that $\|x-y\| \leq \tau_{M,f}$,*

$$\left|1 - \frac{w_{f,q}(x,y)}{D_{M,f}(x,y)}\right| \leq \frac{\kappa}{4f_{\min}} \frac{\|x-y\|}{q-1} + \frac{\|x-y\|^2}{16\mathcal{T}_{M,f}^2} \,. \tag{7}$$

and we denote $\delta_q(\|x-y\|)$ this upper bound. The first term in Equation (7) is to be interpreted as 0 if $q = \infty$ and represents the offset between $w_{f,q}$ and $w_{f,\infty}$, whereas the second term represents the offset between $w_{f,\infty}$ and $D_{M,f}$.

Proposition 3.3 shows that the conformal metric $D_{M,f}$ may be approximated locally by the weights $w_{f,q}$. Note that for fixed resolution q in the r-ball graph, the distortion between the weights and the conformal distances is linear in r. However, by choosing q to be inversely proportional to r, the distortion becomes quadratic in r. The proof for Proposition 3.3 is provided in [17, Appendix B.3]. We now introduce the polygonal approximation of the conformal metric as the metric of the weighted graph built on X.

▶ **Definition 3.4.** *Consider a point cloud $X \subset \mathbb{R}^N$, function $f : \mathbb{R}^N \to \mathbb{R}_+^*$ and parameters $r > 0$ or $k \geq 1$ and $q \in \{2, \dots, \infty\}$. The polygonal metric associated with these parameters is defined between two points $x, y \in \mathbb{R}^N$ as*

$$\widehat{D}_{X,f}(x,y) \overset{def}{=} \min_{(x_0,\dots,x_K)} \sum_{k=0}^{K-1} w_{f,q}(x_k, x_{k+1}) \tag{8}$$

where the minimum is taken over the set of paths $(x_0, \dots, x_K)$ such that $x_0 = x$ and $x_k = y$ in the graph G that is chosen either as the r-ball graph $G_r(X \cup \{x,y\})$ or the k-NN graph $\mathcal{G}_k(X \cup \{x,y\})$.

The choice of a ball graph or a NN graph along with the threshold r or k and the resolution q of the weights are left implicit when writing $\widehat{D}_{X,f}$ to avoid heavy notations. Note that when the endpoints x and y do not belong to the point cloud, the distance $\widehat{D}_{X,f}(x,y)$ is computed by adding them to the graph. As a result, $\widehat{D}_{X,f}$ induces a distance over X but not over M. Indeed, when considering endpoints outside X, $\widehat{D}_{X,f}$ may not satisfy the triangular inequality due to the set of possible paths depending on the endpoints.

3.2 Metric approximation

To evaluate how efficient is the approximation of a metric, the error between the true distances and their estimation may be measured using either of the multiplicative loss functions

$$\ell_{\infty,M}(D', D) \overset{\text{def}}{=} \sup_{x \neq y \in M} \left|\frac{D'(x,y) - D(x,y)}{D'(x,y) \vee D(x,y)}\right| \quad \text{and} \quad l_{\infty,M}(D'|D) \overset{\text{def}}{=} \sup_{x \neq y \in M} \left|1 - \frac{D'(x,y)}{D(x,y)}\right| .$$

In the case where D' takes infinite values – which happens for instance when D' is the metric of a non-connected graph – we let $\ell_{\infty,M}(D', D) = 1$ and $l_{\infty,M}(D'|D) = +\infty$. The inequality

$$\ell_{\infty,M}(D', D) \leq l_{\infty,M}(D'|D) \leq \frac{\ell_{\infty,M}(D', D)}{1 - \ell_{\infty,M}(D', D)} \tag{9}$$

holds in general, so that both losses are equivalent. The loss $\ell_{\infty,M}$ is however easier to manipulate in some situations as it is symmetric and upper bounded by 1. Thanks to the regularity of geodesics stated by Proposition 2.4 and the local approximation of the conformal metric by the weights $w_{f,q}$ stated by Proposition 3.3, we are able to show that the conformal metric $D_{M,f}$ is approximated by the polygonal metric from Definition 3.4 using the r-ball graph.

▶ **Theorem 3.5.** *Let $X \subset M$ be a point cloud, $r > 0$ and $q \in \{2, \dots, \infty\}$ two parameters. Assume that $4d_H(M, X) \leq r \leq \mathcal{T}_{M,f}$. Then the approximation $\widehat{D}_{X,f}$ defined in Definition 3.4 using the r-ball graph with resolution q satisfies*

$$l_{\infty,M}\big(\widehat{D}_{X,f} | D_{M,f}\big) \leq \frac{1}{32\mathcal{T}_{M,f}} \frac{r}{q-1} + \frac{r^2}{8\mathcal{T}_{M,f}^2} + 56\frac{d_H(M,X)^2}{r^2} \ . \tag{10}$$

The first term in Equation (10) *is to be interpreted as* 0 *if* $q = \infty$.

Let us describe the reasoning behind Theorem 3.5. Given $x \neq y \in X$, paths in $G_r(X)$ cannot create significant shortcuts outside M as $r \leq \tau_{M,f}$. This is illustrated by Proposition 3.3 from which it can be deduced that

$$\widehat{D}_{X,f}(x,y) \geq \big(1 - \delta_q(r)\big) D_{M,f}(x,y) \tag{11}$$

and, assuming that $\|x - y\| \leq r$,

$$\widehat{D}_{X,f}(x,y) \leq \big(1 + \delta_q(r)\big) D_{M,f}(x,y) \ . \tag{12}$$

As for when $\|x - y\| > r$, consider a geodesic $\gamma \in \Gamma^\star_{M,f}(x,y)$. Decompose γ into sections of length at most $r - 2d_{\mathrm{H}}(M,X)$ and for each intermediate point select a point in X at distance at most $d_{\mathrm{H}}(M,X)$. This process draws a polygonal path in $G_r(X \cup \{x,y\})$ as it uses edges of length at most r. Summing the approximation error from Proposition 3.3 between each edge of this path and the corresponding section of γ eventually yields the upper bound

$$\widehat{D}_{X,f}(x,y) \leq \left(1 + \delta_q(r) + c_1 \frac{d_{\mathrm{H}}(M,X)}{\mathcal{T}_{M,f}} + c_2 \frac{d_{\mathrm{H}}(M,X)^2}{r^2}\right) D_{M,f}(x,y) \tag{13}$$

when $\|x - y\| > r$ and where c_1 and c_2 are universal constants. The positive reach of γ and Lemma 2.5 play a crucial role in obtaining terms of order $d_{\mathrm{H}}(M,X)/\mathcal{T}_{M,f}$ and $d_{\mathrm{H}}(M,X)^2/r^2$ instead of $d_{\mathrm{H}}(M,X)/r$. Together, Equations (11)–(13) eventually imply Equation (10). The details of this reasoning are provided in [17, Appendix B.4]. The upper bound in Equation (10) is made proportional to $d_{\mathrm{H}}(M,X)$ by setting

$$r \asymp \sqrt{\mathcal{T}_{M,f}\ d_{\mathrm{H}}(M,X)} \quad \text{and} \quad q \asymp \sqrt{\frac{\mathcal{T}_{M,f}}{d_{\mathrm{H}}(M,X)}} \ .$$

In the case of the induced metric with $f = 1$, the first term in the right-hand side of Equation (10) disappears, and the error is of order $r^2/\tau_M^2 + d_{\mathrm{H}}(M,X)^2/r^2$ similarly to the one obtained in [3]. In the general conformal case, we keep this magnitude of error by setting the resolution q to be at least of order $1/r$. If it is not possible to evaluate f outside the point cloud X, setting $q = 2$ instead yields an error of order $r/\mathcal{T}_{M,f} + d_{\mathrm{H}}(M,X)^2/r^2$ which is slightly worse. A similar upper bound was achieved in [2] for the induced metric and the term linear in r was due to the geodesically smooth assumption used in the paper being slightly weaker than a positive reach assumption. In particular, the latter allows for an efficient local estimation in the form of Equation (12). The main takeaway of Theorem 3.5 is that the crucial assumption to achieve an approximation error proportional to $d_{\mathrm{H}}(M,X)$ is the positive reach. In particular, the domain need not be $\mathcal{C}^2$, or even a submanifold.

3.3 Unknown conformal factor

In general, f may not be known and needs to be estimated from the data. The polygonal metric $\widehat{D}_{X,g}$ from Definition 3.4 can be defined for any function $g : \mathbb{R}^N \to \mathbb{R}_+^*$ without the need of Lipschitz and lower bound assumptions. Then, the approximation error between $\widehat{D}_{X,g}$ and $D_{M,f}$ is upper bounded by the sum of the error w.r.t. the conformal factor and the error w.r.t. the domain.

▶ **Lemma 3.6.** *Let $f : \mathbb{R}^N \to \mathbb{R}_+$ a function lower bounded by $f_{\min}$, $X \subset M$ a point cloud and $g : \mathbb{R}^N \to \mathbb{R}_+^*$ such that $\|g - f\|_\infty \leq \frac{1}{2} f_{\min}$. Then*

$$\ell_{\infty,M}\big(\widehat{D}_{X,g}, D_{M,f}\big) \leq \frac{2}{f_{\min}}\|g - f\|_\infty + 2\ell_{\infty,M}\big(\widehat{D}_{X,f}, D_{M,f}\big) \ . \tag{14}$$

This result holds regardless of the type of graph, threshold and resolution as long as they are shared between $\widehat{D}_{X,f}$ and $\widehat{D}_{X,g}$.

The proof of Lemma 3.6 consists in straightforward computations and is deferred to [17, Appendix B.5]. As a consequence, in the context of estimation of the domain M by a point cloud X_n as in Section 4 and of the conformal factor f by a function $f_n : \mathbb{R}^N \to \mathbb{R}_+$, the rate of convergence of $\widehat{D}_{X_n,f_n}$ towards $D_{M,f}$ is the slowest rate of convergence among that of $\widehat{D}_{X_n,f}$ towards $D_{M,f}$ and that of f_n towards f. Recall that f_n is not required to satisfy the Lipschitz and lower bounded assumptions for the polygonal metric $\widehat{D}_{X_n,f_n}$ to be defined and for Lemma 3.6 to hold. It may also be defined only over X_n if the resolution is set to $q = 2$.

4 Estimation from random samples

In this section we transpose Theorem 3.5 to a probabilistic setup where the point cloud X is replaced with a random point cloud X_n consisting of n i.i.d. samples from a probability measure μ with support M. The following assumptions on μ are needed to ensure that X_n covers M efficiently as n grows, allowing to deduce explicit convergence rates for the estimator $\widehat{D}_{X_n,f}$ from Theorem 3.5.

▶ **Definition 4.1.** *Consider a probability measure μ with support $M \subset \mathbb{R}^N$ and $d \geq 2$.*

- *The measure μ is d-standard with lower constant $c_\mu > 0$ if for all $x \in M$ and $r > 0$,*

$$\mu\big(B(x,r)\big) \geq c_\mu r^d \wedge 1$$

  where $B(x,r)$ denotes the open ball centered at x with radius r.

- *The measure μ is d-Ahlfors with lower and upper constants $c_\mu > 0$ and $C_\mu > 0$ if for all $x \in M$ and $r > 0$,*

$$c_\mu r^d \wedge 1 \leq \mu\big(B(x,r)\big) \leq C_\mu r^d \ .$$

If μ is d-standard with lower constant c_μ, we also denote $L_\mu = c_\mu^{-1/d}$ which is related to the size of the support. The definition ensures that $\mu\big(\mathrm{B}(x, L_\mu)\big) = 1$ for all $x \in M$, hence $L_\mu \geq \mathrm{diam}(M)$. Moreover, d acts as the intrinsic dimension of the support M of μ. For instance, any measure with a density ρ w.r.t. the volume measure of a submanifold of $\mathbb{R}^N$ of dimension d is d-Ahlfors if ρ is bounded above and below. The d-standard assumption ensures that the random point cloud X_n converges to M in Hausdorff distance. Ahlfors regularity is a stronger assumption and is needed to show the equivalence between ball graphs and NN graphs in Proposition 4.4.

4.1 Convergence of the ball graph estimator

We first discuss the case of a ball graph estimator. It is known that if μ is d-standard, then $d_{\mathrm{H}}(M, X_n)$ is of order $L_\mu(\log(n)/n)^{1/d}$ at most, see [17, Appendix A.2]. This convergence combined with Theorem 3.5 allows to derive convergence rates for the estimation of $D_{M,f}$.

▶ **Theorem 4.2.** *Assume that X_n is the result of n i.i.d. samples from a d-standard probability measure μ with support M. Consider the estimator $\widehat{D}_{X_n,f}$ from Definition 3.4 using the r-ball graph with resolution q where r and q are specified below. Then there exists a constant n_0 depending on L_μ, $\mathcal{T}_{M,f}$ and d such that for all $n \geq n_0$ the following holds.*

- *If $r = 8\sqrt{L_\mu \mathcal{T}_{M,f}} \left(\frac{\log(n)}{n}\right)^{\frac{1}{2d}}$ and $q \geq 1 + 4\frac{\mathcal{T}_{M,f}}{r}$, then*

$$\mathbb{E}\left[\ell_{\infty,M}\big(\widehat{D}_{X_n,f}, D_{M,f}\big)\right] \leq 16\frac{L_\mu}{\mathcal{T}_{M,f}}\left(\frac{\log(n)}{n}\right)^{\frac{1}{d}} . \tag{15}$$

- *If $r = 8L_\mu^{2/3}\mathcal{T}_{M,f}^{1/3}\left(\frac{\log(n)}{n}\right)^{\frac{2}{3d}}$ and $q = 2$, then*

$$\mathbb{E}\left[\ell_{\infty,M}\big(\widehat{D}_{X_n,f}, D_{M,f}\big)\right] \leq 8\left(\frac{L_\mu}{\mathcal{T}_{M,f}}\right)^{\frac{2}{3}}\left(\frac{\log(n)}{n}\right)^{\frac{2}{3d}} . \tag{16}$$

The first case in Theorem 4.2 includes $q = \infty$. There is no use setting q to be greater than the indicated threshold of order $\mathcal{T}_{M,f}/r$, as the term in the upper bound depending on q becomes negligible past this threshold. Note that the optimal choice for r and q requires the knowledge of L_μ, $\mathcal{T}_{M,f}$ and most importantly d. Theorem 4.2 is stated with these choices of parameters to highlight the optimal theoretical dependency in these constants. For practical purposes, the knowledge of L_μ and $\mathcal{T}_{M,f}$ is not actually needed as, in general, any choice of $r \asymp (\log(n)/n)^{1/2d}$ and $q \gtrsim 1/r$ yields a convergence rate of $(\log(n)/n)^{1/d}$. Likewise, any choice of $r \asymp (\log(n)/n)^{2/3d}$ and $q = 2$ yields a convergence rate of $(\log(n)/n)^{2/3d}$. It is not possible however to get rid of the dependency in d for the range parameter r without altering the convergence rate. To circumvent this issue, one may use the NN graph instead, for which the optimal choice of the parameter is shown in Section 4.2 to be $k = \sqrt{n}$ which does not depend on d. Theorem 4.2 is derived directly from Theorem 3.5 and the convergence of X_n to M in Hausdorff distance. The precise computations are deferred to [17, Appendix C.1]. In the case of the induced metric, the weight function $w_{f,q}$ becomes the Euclidean distance regardless of the resolution and $\mathcal{T}_{M,f}$ is replaced with τ_M.

▶ **Corollary 4.3.** *Under the same assumptions as in Theorem 4.2 and when $f = 1$, there exists a constant n_0 depending on L_μ, τ_M and d such that for all $n \geq n_0$, setting $r = 8\sqrt{L_\mu \tau_M}(\log(n)/n)^{1/2d}$, the estimator $\widehat{D}_{X_n,1}$ satisfies*

$$\mathbb{E}\left[\ell_{\infty,M}\big(\widehat{D}_{X_n,1}, D_M\big)\right] \leq 16\frac{L_\mu}{\tau_M}\left(\frac{\log(n)}{n}\right)^{\frac{1}{d}} .$$

Recall that in the case of a submanifold of class $\mathcal{C}^k$ with $k \geq 2$ and dimension d, the optimal convergence rate for the estimation of the induced metric is $n^{-k/d}$ [1]. Corollary 4.3 extends the upper bound to any set of positive reach with the convergence rate $n^{-1/d}$.

4.2 Convergence of the nearest-neighbors graph estimator

We now treat the case of a NN graph estimator, by observing that for adequate choice of r and k, the r-ball graph and k-NN graph are similar. Indeed, assuming that μ is d-Ahlfors regular, a ball of radius r is expected to contain between $c_\mu n r^d$ and $C_\mu n r^d$ points from X_n.

▶ **Proposition 4.4.** *Let μ be a d-Ahlfors measure as defined in Definition 4.1 and X_n a point cloud sampled i.i.d. from μ. Let $k \geq 1$, $\varepsilon \in (0,1)$ and define*

$$r_- = \left((1-\varepsilon)\frac{k}{C_\mu(n-1)}\right)^{\frac{1}{d}} \quad \text{and} \quad r_+ = \left(\frac{1}{1-\varepsilon}\frac{k}{c_\mu(n-1)}\right)^{\frac{1}{d}} .$$

Then with probability at least $1 - 2ne^{-\varepsilon^2 k/2}$*, the* k*-NN graph* $\mathcal{G}_k(X_n)$ *is enclosed between two ball graphs* $G_{r_-}(X_n)$ *and* $G_{r_+}(X_n)$*, that is*

$$\mathbb{P}\left(G_{r_-}(X_n) \subset \mathcal{G}_k(X_n) \subset G_{r_+}(X_n)\right) \geq 1 - 2n\exp\left(-\frac{\varepsilon^2}{2}k\right)$$

where the inclusion $\subset$ *refers to the inclusion of edge sets.*

Proposition 4.4 shows that if r and k are chosen such that $k \asymp nr^d$, the k-NN graph and the r-ball graph are very similar with high probability. This result stems from a common intuition and a detailed proof is provided in [17, Appendix A.3] for completeness. Theorem 4.2 is then extended to k-NN graphs by choosing $k \asymp \sqrt{n\log(n)}$, which does not depend on d.

▶ **Theorem 4.5.** *Assume that* X_n *is the result of* n *i.i.d. samples from a* d*-Ahlfors probability measure* μ *with support* M*. Consider the estimator* $\widehat{D}_{X_n,f}$ *from Definition 3.4 using the* k*-NN graph with parameters* $k = \lceil\sqrt{n\log(n)}\,\rceil$ *and* $q = \lceil n^{1/4}\rceil$*. Then there exists a constant* n_0 *depending on* L_μ*,* $\mathcal{T}_{M,f}$ *and* d *such that for all* $n \geq n_0$ *the following holds.*

$$\mathbb{E}\left[\ell_{\infty,X_n}\left(\widehat{D}_{X_n,f}, D_{M,f}\right)\right] \leq C\left(\frac{\log(n)}{n}\right)^{\frac{1}{d}}$$

where C *is a constant depending on* L_μ *and* $\mathcal{T}_{M,f}$*.*

Notice that in Theorem 4.5 the loss is over X_n instead of M like in Theorem 4.2. The same result could be stated over M, although the proof would be more tedious and is therefore omitted here. Recall that no knowledge on either d, c_μ, C_μ, τ_M, κ or $f_{\min}$ is necessary to compute the estimator, so that it can be used in practice. Setting the resolution q to be $(n/\log(n))^{1/2d}$ would be sufficient to achieve the same upper bound, however this choice requires the knowledge of d. On the other hand, when setting the resolution to $q = 2$ the optimal choice of k can be shown similarly to be $k = n^{1/3}\log(n)^{2/3}$ and to yield a convergence rate of order $(\log(n)/n)^{2/3d}$. Theorem 4.5 is a direct consequence of Theorem 4.2 and Proposition 4.4 and the precise computations are provided in [17, Appendix C.2]. Regarding the case of the induced metric, the same statement as in Corollary 4.3 holds for NN graphs. Namely, the estimator $\widehat{D}_{X_n,1}$ defined over the k-NN graph with $k = \lceil\sqrt{n\log(n)}\,\rceil$ converges to D_M at rate $(\log(n)/n)^{1/d}$.

Let us now discuss the algorithmic complexity of the k-NN estimator with resolution q. In general, building the k-NN graph over a point cloud X of size n is done in $\mathcal{O}(n^2N)$ time when the ambient dimension N is large. Now, consider two points $x, y \in \mathbb{R}^N$ and assume that f can be evaluated at any point with cost c_f. Computing the edges of the graph $\mathcal{G}_k(X \cup \{x,y\})$ takes $\mathcal{O}(nkqc_f)$ time as the amount of edges in the graph is $\mathcal{O}(nk)$ and the weight of each edge uses q evaluations of f. Finally, computing the infimum that defines $D_{X,f}(x,y)$ in Definition 3.4 using Dijkstra's algorithm takes $\mathcal{O}(n\log(n) + nk)$ time. Overall, if $k = \lceil\sqrt{n\log(n)}\,\rceil$ and $q = \lceil n^{1/4}\rceil$, the time complexity is $\mathcal{O}(n^2N + n^{7/4}\log(n)^{1/2}c_f)$.

4.3 Minimax lower bound

We now study the worst case performance of any estimator of the induced metric D_M.

▶ **Theorem 4.6.** *Denote* $\mathfrak{D}_n$ *the set of all estimators* $\widehat{D}$ *of the induced metric based on* n *samples, that given any point cloud* X *of* n *points in* $\mathbb{R}^N$ *provides a function* $\widehat{D}_X : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}_+$*. Let* $2 \leq d \leq N$*,* $L, \tau > 0$ *and* $\mathcal{M}(d, L, \tau)$ *be the set of all* d*-standard measures* μ *with* $L_\mu \leq L$ *and support* $M_\mu \subset \mathbb{R}^N$ *that has positive reach lower bounded by* τ*. Then there exists two constants* $C > 0$ *and* n_0 *depending on* d*,* L *and* τ *such that for all* $n \geq n_0$*,*

$$\inf_{\widehat{D}\in\mathfrak{D}_n} \sup_{\mu\in\mathcal{M}(d,L,\tau)} \mathbb{E}_{X\sim\mu^{\otimes n}}\left[\ell_{\infty,M_\mu}\left(\widehat{D}_X, D_{M_\mu}\right)\right] \geq C\left(\frac{1}{n}\right)^{\frac{1}{d-1/2}} . \tag{17}$$

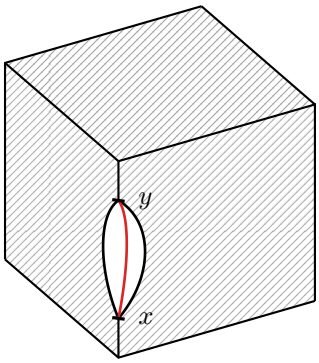


Figure 2 Carving an edge of the cube in $\mathbb{R}^3$.

Notice that Theorem 4.6 only addresses the case of the induced metric. This is not a loss of generality and in fact highlights the fact that the conformal change does not make the problem any harder, as we have established in Section 2 that geodesics have the same regularity as in the case of the induced metric. Given any other function f satisfying the assumptions for a conformal metric, the same lower bound may be obtained with a similar reasoning as what follows, albeit with more technicalities. The rate $n^{-1/(d-1/2)}$ is faster than the rate $n^{1/d}$ obtained in Theorems 4.2 and 4.5, although the difference becomes negligible when d is large. The nature of the minimax convergence rate remains therefore open. Under stronger assumptions on the domain, this question is already solved – see [1] – which we discuss in Section 4.4.

Theorem 4.6 is based on Le Cam's method [19], for which a statement adapted to our setup is given in [17, Lemma C.1]. The method consists in finding two measures μ_1 and μ_2 in $\mathcal{M}(d, L, \tau)$ that are at most $1/n$ apart in total variation distance and such that the relative difference between D_{M_1} and D_{M_2} is of order at least $n^{-1/(d-1/2)}$.

Consider μ_1 the uniform probability on the cube $M_1 = [-\alpha L, \alpha L]^d \times \{0\}^{N-d} \subset \mathbb{R}^N$ where $\alpha > 0$. Let $0 < \varepsilon \leq \alpha L \wedge \tau$ and consider M_2 the result of removing from M_1 its intersection with a ball of radius τ centered at $(0, t, t, \dots, t)$ for some $t > \alpha L$ such that the ball intersects the edge from $(-\alpha L, \alpha L, \alpha L, \dots, \alpha L)$ to $(\alpha L, \alpha L, \alpha L, \dots, \alpha L)$ at two points x and y that are 2ε apart, as pictured in Figure 2. Denote μ_2 the uniform probability over M_2, which has reach exactly τ due to the carved area. By choosing α small enough, which depends only on d, μ_1 and μ_2 are d-standard with lower constant L^{-d}. Then μ_1 and μ_2 both belong to $\mathcal{M}(d, L, \tau)$. The volume of the carved area is of order $\varepsilon(\varepsilon^2)^{d-1}$ as it spans a length 2ε between x and y and a length of order ε^2 for every other dimension of the cube. This implies that the total variation distance between μ_1 and μ_2 is of order ε^{2d-1}. Moreover, The distance from x to y goes from 2ε in M_1 to $2\tau \arcsin(\varepsilon/\tau)$ in M_2, following the red arc of radius τ in Figure 2. This implies that the relative difference between both distances is of order ε^2. Choosing $\varepsilon \asymp n^{-1/2d-1}$ so that the total variation distance is at most $1/n$ eventually yields the desired bound. This reasoning is detailed in [17, Appendix C.3] and the reason why it fails to achieve a bound of order $n^{-1/d}$ is discussed more in depth in the full version [17].

4.4 Case of a smooth manifold

Recall that the minimax convergence rate for the induced metric of a $\mathcal{C}^k$ manifold of dimension d is $n^{-k/d}$ [1]. The methods that achieve such rates are however not computationally feasible in practice as they rely on manifold reconstruction via non-discrete sets. For instance, the

optimal convergence rate in the $\mathcal{C}^2$ case was achieved by [3] using the tangential Delaunay complex. Precisely, these methods build a $n^{-k/d}$-approximation of the manifold w.r.t. the Hausdorff distance, then state that the induced metric over this reconstruction is an estimation of the original metric with an error proportional to the Hausdorff distance.

In order to get a concrete estimator, one may use such manifold reconstruction, then sample a fine $n^{-k/d}$-net over it which approximates the original domain with the same Hausdorff error. Our work then implies that the polygonal metric over this net would be an estimation of the original metric with minimax optimal error of order $n^{-k/d}$. Such construction would however be very costly as the point cloud size would grow from n to n^k.

5 Practical examples of conformal factors

We discuss two examples of conformal factors associated with a measure from the literature.

5.1 Density as a conformal factor

If μ is a measure with density ρ w.r.t. the volume measure over a submanifold M, the conformal change via $f = \rho^{-\beta}$ for some parameter $\beta > 0$ is sometimes referred to as the *Fermat distance* due to the parallel with the Fermat principle in optics – that may however be observed with any conformal change. This metric has been applied to various topological data analysis and learning problems, e.g., in [11, 12], as it tends to accentuate features and disparities in the data. Moreover, the metric can be estimated in a simple fashion by using the same overall reasoning as ours but using weights of the form $w(x, y) = \|x - y\|^\alpha$ where $\alpha > 1$ is a parameter depending on β and d [13, 14]. However, this kind of estimation does not feature any known convergence rate. Using our estimator instead provides an alternative method with quantitative guarantees, granted that an estimator of the density is available. To this extent, density estimation is a well-studied problem with many propositions in the literature. For instance, [4] provides a kernel density estimator ρ_n that converges in L^p norm towards ρ at a rate of $n^{-1/d+1}$ under our assumptions and provided that M is a $\mathcal{C}^1$ submanifold and that ρ is also $\mathcal{C}^1$. Under these assumptions, setting $f_n = \rho_n^{-\beta}$ yields a convergence rate of $n^{-1/d}$ according to Lemma 3.6, at the cost of a more complex computation than the usual discrete Fermat distance studied in [13].

5.2 Distance-to-measure as a conformal factor

In general, given a parameter $m \in (0, 1)$, the *distance-to-measure* $d_{\mu,m} : \mathbb{R}^N \to \mathbb{R}_+$ introduced in [8] is defined for any measure μ over $\mathbb{R}^N$. It is 1-Lipschitz and lower bounded by a positive value as long as μ has no atom, hence satisfies the assumption for our work. A slightly different setup where paths are allowed to leave the domain under a specific constraint is studied in [18], where it is argued that this metric should behave similarly to the conformal metric associated with the density but with more stability w.r.t. the measure. The distance-to-measure is shown [9] to be estimated from n i.i.d. samples of the underlying measure with convergence rate $n^{-1/2}$, which is faster than $n^{-1/d}$ hence does not impact the convergence speed of the estimator of the conformal metric according to Lemma 3.6.

6 Conclusion

In this work, we have shown that under a reach assumption on the domain and Lipschitz lower bounded assumptions on the conformal change, the conformal metric has the same regularity as the induced metric in the sense of geodesics having positive reach. We have

also shown that positive reach is a sufficient assumption to ensure metric estimation from a polygonal metric with error proportional to the Hausdorff distance between the point cloud and the original domain. This leads to a convergence rate of order $n^{-1/d}$ for the estimation of the conformal metric using n i.i.d. samples from a d-standard measure, which in particular applies to the induced metric of any set with positive reach.

References

1 Eddie Aamari, Clément Berenfeld, and Clément Levrard. Optimal reach estimation and metric learning. *The Annals of Statistics*, 51(3):1086–1108, 2023-06. doi:10.1214/23-AOS2281.

2 Catherine Aaron and Olivier Bodart. Convergence rates for estimators of geodesic distances and Frèchet expectations. *Journal of Applied Probability*, 55(4):1001–1013, 2018-12. doi:10.1017/jpr.2018.66.

3 Ery Arias-Castro, Adel Javanmard, and Bruno Pelletier. Perturbation Bounds for Procrustes, Classical Scaling, and Trilateration, with Applications to Manifold Learning. *Journal of Machine Learning Research*, 21(1):498–534, 2020. URL: http://jmlr.org/papers/v21/18-720.html.

4 Clément Berenfeld and Marc Hoffmann. Density estimation on an unknown submanifold. *Electronic Journal of Statistics*, 15(1):2179–2223, 2021-01. doi:10.1214/21-EJS1826.

5 Mira Bernstein, Vin Silva, John Langford, and Joshua Tenenbaum. Graph Approximations to Geodesics on Embedded Manifolds, 2000-12-20. URL: https://api.semanticscholar.org/CorpusID:10751942.

6 Jean-Daniel Boissonnat, André Lieutier, and Mathijs Wintraecken. The reach, metric distortion, geodesic convexity and the variation of tangent spaces. *Journal of Applied and Computational Topology*, 3(1):29–58, 2019-06-01. doi:10.1007/s41468-019-00029-8.

7 Dmitri Burago, Yuri Burago, and Sergei Ivanov. *A Course in Metric Geometry*, volume 33 of *Graduate Studies in Mathematics*. American Mathematical Society, 2001-06-12. doi:10.1090/gsm/033.

8 Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric Inference for Measures based on Distance Functions. *Foundations of Computational Mathematics*, 11(6):733, 2011. doi:10.1007/s10208-011-9098-0.

9 Frédéric Chazal, Pascal Massart, and Bertrand Michel. Rates of convergence for robust geometric inference. *Electronic Journal of Statistics*, 10(2):2243–2286, 2016-01. doi:10.1214/16-EJS1161.

10 Herbert Federer. Curvature measures. *Transactions of the American Mathematical Society*, 93(3):418–491, 1959. doi:10.1090/S0002-9947-1959-0110078-1.

11 Ximena Fernández, Eugenio Borghini, Gabriel Mindlin, and Pablo Groisman. Intrinsic persistent homology via density-based metric learning. *Journal of Machine Learning Research*, 24(1):75:3341–75:3382, 2023-01-01. URL: https://jmlr.org/papers/v24/21-1044.html.

12 Nicolas Garcia Trillos, Anna Little, Daniel McKenzie, and James M. Murphy. Fermat Distances: Metric Approximation, Spectral Convergence, and Clustering Algorithms. *Journal of Machine Learning Research*, 25(1):176:8331–176:8395, 2024-01-01. URL: http://jmlr.org/papers/v25/23-0939.html.

13 Pablo Groisman, Matthieu Jonckheere, and Facundo Sapienza. Nonhomogeneous Euclidean first-passage percolation and distance learning. *Bernoulli*, 28(1):255–276, 2022-02. doi:10.3150/21-BEJ1341.

14 Sung Jin Hwang, Steven B. Damelin, and Alfred O. Hero Iii. Shortest path through random points. *The Annals of Applied Probability*, 26(5):2791–2823, 2016-10. doi:10.1214/15-AAP1162.

15 André Lieutier and Mathijs Wintraecken. Manifolds of positive reach, differentiability, tangent variation, and attaining the reach, 2024-12-06. doi:10.48550/arXiv.2412.04906.

16 E. J. McShane. Extension of range of functions. *Bulletin of the American Mathematical Society*, 40(12):837–842, 1934. doi:10.1090/S0002-9904-1934-05978-0.

17 Jérôme Taupin. Estimation of Conformal Metrics.

18 Jérôme Taupin and Frédéric Chazal. Fermat Distance-to-Measure: A robust Fermat-like metric, 2025-04-03. arXiv:2504.02381.

19 Bin Yu. Assouad, Fano, and Le Cam. *Research Papers in Probability and Statistics*, 1997. doi:10.1007/978-1-4612-1880-7_29.

Simplicial Approximation to CW Complexes with Spherical Delaunay Triangulations

Raphaël Tinarrage
Institute of Science and Technology Austria, Klosterneuburg, Austria

Abstract

Simplicial approximation provides a framework for constructing simplicial complexes that are homotopy equivalent to a given manifold, provided a CW structure is explicitly known. However, its conventional implementation quickly becomes intractable on a computer: barycentric subdivision produces poorly shaped simplices, and the star condition introduces many vertices. To address these limitations, this article develops a subdivision scheme based on spherical Delaunay triangulations, which attains better refinement properties than barycentric subdivisions. Moreover, the star condition is reframed as two independent problems, one geometric and the other combinatorial, respectively tackled in the language of locally equiconnected spaces and the list homomorphism problem, allowing an exponential reduction in the number of vertices. Via a prototype implementation, we obtain simplicial complexes homotopy equivalent to Grassmannians and Stiefel manifolds up to dimension 5.

2012 ACM Subject Classification Mathematics of computing → Mesh generation; Mathematics of computing → Combinatoric problems; Mathematics of computing → Algebraic topology; Mathematics of computing → Mathematical software performance

Keywords and phrases Triangulation of manifolds, Simplicial approximation, CW complexes, Delaunay complexes, List homomorphism problem, Topological Data Analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.93

Related Version *Full Version*: `https://arxiv.org/abs/2112.07573`

Supplementary Material *Software*: `https://doi.org/10.5281/zenodo.19251455`

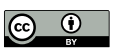
1 Introduction

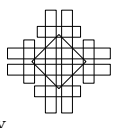
1.1 Topology software

In computational topology, a popular way to represent a topological space is via a simplicial complex. Once a space is triangulated, it can be explored algorithmically, and a range of homotopy invariants can be evaluated [36, 37, 38, 69, 39, 64, 65, 66, 112, 90]. More generally, triangulations open the door to many further developments: they allow us to test conjectures and discover new properties [16, 15], they serve as a benchmark for comparing software [14, 9, 108] and they lay the foundation for new data analysis techniques [103, 106, 59, 111].

Over the past two decades, software for computations on simplicial complexes has advanced substantially. Existing libraries span a wide range of goals, from algebra to geometry and data analysis. They have underpinned numerous concrete advances: proofs and counterexamples in 3-manifold topology with `Regina` and `SnapPy` [35, 48]; large-scale enumerations that tested conjectures with `BISTELLAR` and `Twister` [18, 13]; new homotopy and cohomology computations in `Kenzo` and `HAP` [50, 56]; and persistent homology pipelines that surfaced new properties of datasets with `GUDHI`, `Ripser`, and `TTK` [91, 11, 110], among many others.

On the other hand, computational topology lacks *explicit* examples of triangulated manifolds, as noted in [16, 66, 9, 108]. By explicit, we mean stored on a computer as a list of simplices, or obtainable in reasonable time by an implemented algorithm. This is especially striking in dimension 4 and above, as summarized in Table 1, which collects known


© Raphaël Tinarrage;
licensed under Creative Commons License CC-BY 4.0



triangulations of certain classical manifolds. Apart from the real and complex projective spaces, most triangulations are "accidental", i.e., arising from special homeomorphisms with known spaces. The aim of this article is to develop an algorithm for triangulating new spaces.

Table 1 Known explicit triangulations of a selection of manifolds.

Space	Known cases	References / Known identifications
Real projective space $\mathbb{R}P^n$	$n \geq 1$	[83, 8, 4]
Complex projective space $\mathbb{C}P^n$	$n \geq 1$	[84, 102, 6, 7, 101, 49]
Special orthogonal group $\mathrm{SO}(n)$	$n \leq 4$	$\mathrm{SO}(3) \cong \mathbb{R}P^3$, $\mathrm{SO}(4) \cong S^3 \times \mathrm{SO}(3)$
Special unitary group $\mathrm{SU}(n)$	$n \leq 2$	$\mathrm{SU}(2) \cong S^3$
Unitary group $\mathrm{U}(n)$	$n = 1$	$\mathrm{U}(1) \cong S^1$
Real Stiefel manifold $\mathcal{V}(d, n)$	$d = 1$ or $n \leq 4$	$\mathcal{V}(1, n) \cong S^{n-1}$, $\mathcal{V}(d-1, d) \cong \mathrm{SO}(d)$
Real Grassmannian $\mathcal{G}(d, n)$	$d = 1$ or $d = n - 1$	$\mathcal{G}(1, n) \cong \mathcal{G}(n-1, n) \cong \mathbb{R}P^{n-1}$

1.2 Related work

Traditionally, explicit triangulations of manifolds are obtained by two main approaches. The first is combinatorial and relies on specific descriptions of the manifolds under consideration. For example, small triangulations of $\mathbb{R}P^n$ have recently been constructed by exploiting its realization from a symmetric polytope [4]. Topological properties can also guide the enumeration of combinatorial manifolds, as in the work of Lutz [87, 107]. However, combinatorial complexity grows rapidly with the dimension. Thus, in dimensions 3 and 4, more structured constructions are preferred, based on layered triangulations [76, 75, 77], Dehn fillings [52], Heegaard diagrams [57, 74, 58], and Kirby diagrams [33, 34]. The present work follows this perspective by exploiting the *CW structure* of the spaces involved, which is well understood.

Sampling-based methods offer a different viewpoint. A large body of work studies how to reconstruct an embedded manifold $\mathcal{M} \subset \mathbb{R}^n$ from a finite sample $X \subset \mathbb{R}^n$. The goal is usually not to recover a triangulation of $\mathcal{M}$ itself, but rather a simplicial complex with the same homotopy type. A standard construction is the Čech complex. Recovery is guaranteed when the sampling density is sufficiently fine relative to an appropriate condition number, typically the reach of $\mathcal{M}$ [94]. Quantitative refinements use less restrictive geometric quantities, such as the μ-reach [40, 78], weak feature size [41], local feature size [42], convexity defects [5], and convexity radius [61]. In higher dimensions, however, the Čech complex may be prohibitively expensive to compute, and one may instead use the Vietoris–Rips complex, which depends only on pairwise distances. Following the foundational results of Hausmann and Latschev [73, 85], quantitative guarantees have been established in terms of the reach [88, 89], the μ-reach [78], and convexity defects [5]. Other approaches rely on Delaunay complexes [23, 24, 22, 1], witness complexes [104, 70, 25, 26], and metric thickenings [2, 3].

These methods nevertheless remain computationally demanding. Vietoris–Rips complexes are the cheapest to compute, but their guarantees are weaker and they typically require many vertices. Čech complexes present the same difficulty, as they may contain a large number of simplices, potentially of high dimension. In contrast, more refined constructions such as Delaunay complexes require substantial computation time and scale poorly with dimension.

Grassmannians $\mathcal{G}(d, n)$ are a notable example, regarded as difficult to triangulate. Theoretical results indicate that the minimal number of facets increases exponentially with n [68]. Knudson considered triangulating $\mathcal{G}(2, 4)$ by embedding it into $\mathbb{R}^{16}$ and constructing a Vietoris–Rips complex [79]. In practice, the approach amounted to increasing the scale until

obtaining a complex with the expected homology. This proved too costly in memory, and a witness complex was used instead. Although a complex with the correct homology was found, this construction does not satisfy the hypotheses of the available theorems.

1.3 Contributions

In Algorithm 2 we present an implementation of *simplicial approximation to CW complexes*, a framework well established in theory yet overlooked in practice. Given a CW structure on a topological space, the algorithm outputs a *homotopy equivalent* simplicial complex. Although not homeomorphic, such a complex still suffices for many of the applications mentioned above. Crucially, our complexes are equipped with a *point location* routine, making them practical surrogates for the manifold in data-driven applications.

Turning the textbook construction into a practical implementation requires several adjustments; in particular, we aim to keep the resulting complexes as small as possible. To this end, in Section 2, we adopt Delaunay refinement instead of barycentric subdivision; in Section 3, we build a simplicial mapping cone that avoids subdividing the complex; and in Section 4, we substitute the star condition for a more efficient constraint satisfaction problem.

We implemented two versions of the algorithm, using either global or local refinement. Only the first is currently proven to terminate, but the second often produces smaller complexes. In both cases, when the algorithm terminates, the output is correct (see Theorem 13).

The software is provided as supplementary material.[1] The repository includes complexes homotopy equivalent to $\mathcal{G}(2,4)$ and $\mathcal{V}(2,4)$ (of dimensions 4 and 5), which were not available prior to this work. We intend to address higher-dimensional examples in future work by increasing computational resources; the current results were obtained on a personal laptop.

The algorithm is sketched in the next section. In Sections 2–4 we develop the theoretical results required for its implementation. Proofs appear in the full version of the article.

1.4 Overview and discussion of the construction

CW complexes. The theoretical background can be found in Hatcher's book [72, Section 2.C]. A *CW complex* of dimension n is a topological space X endowed with a decomposition $X = X^n \supset X^{n-1} \supset \cdots \supset X^0$ where X^0 is finite and each X^d is equal to a disjoint union

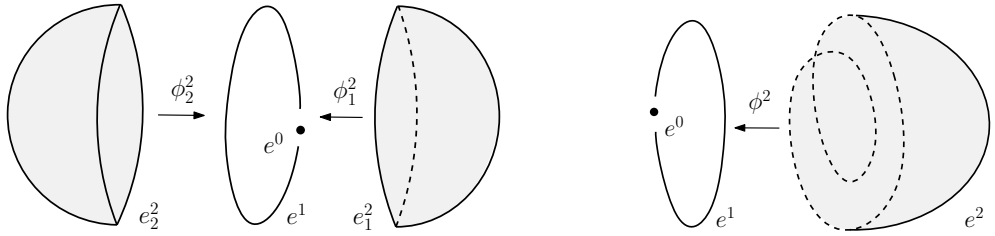
$$X^d = X^{d-1} \coprod_{1 \le i \le m(d)} e_i^d,$$

where each e_i^d, called a *d-cell*, is homeomorphic to an open ball of dimension d, and $m(d)$ is their number. It is required that the homeomorphism extends to the closed ball B^d, yielding a map $\Phi_i^d \colon B^d \to \bar{e}_i^d \subset X^d$, called the *characteristic map*. Its restriction to the boundary of B^d – i.e., the sphere S^{d-1} – is called the *gluing map* and is denoted $\phi_i^d \colon S^{d-1} \to X^{d-1}$. In other words, X^d is homeomorphic to the gluing of d-balls along their boundary on X^{d-1}:

$$X^d \cong X^{d-1} \cup_{\phi_1^d} B^d \cup_{\phi_2^d} B^d \cup \cdots \cup_{\phi_{m(d)}^d} B^d.$$

Figure 1a shows a CW structure on S^2, made of one 0- and 1-cell and two 2-cells. Figure 1b depicts the usual structure on $\mathbb{R}P^2$: one cell per dimension and a gluing map $\phi^2 \colon S^1 \to S^1$ of degree 2. Similarly, all the manifolds in Table 1 admit a well-known CW structure. We refer the interested reader to [92, Section 6] for the classical structure on Grassmannians, [72, Section 3.D] for Stiefel manifolds and orthogonal groups, and [113] for unitary groups.

[1] Fully implemented prototype in Python: `https://github.com/raphaeltinarrage/cw2simp`



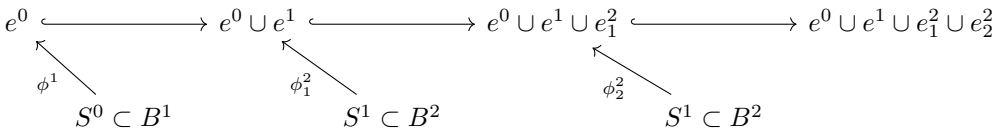
(a) S^2. **(b)** $\mathbb{R}P^2$.

Figure 1 Examples of CW structures on the sphere S^2 (four cells in total) and the projective plane $\mathbb{R}P^2$ (one cell per dimension). If only one d-cell is attached, we omit the subscript in e_i^d.

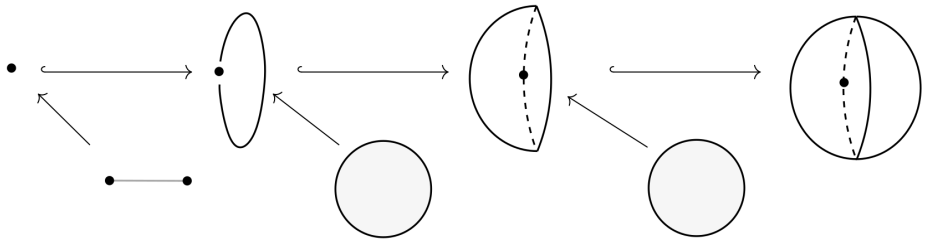
Sketch of the approach. One can "convert" a CW complex into a simplicial complex by gluing *simplicial balls* instead of cells; the idea is sketched in Figure 2. The construction proceeds inductively on the dimension. Suppose we have already built a simplicial complex L^d homotopy equivalent to the d-skeleton X^d; we denote by $|L^d|$ its geometric realization. For each $(d+1)$-cell e_i^{d+1}, we perform the following steps:

- We apply *simplicial approximation* (see below) to the gluing map $\phi_i^{d+1}\colon S^d \to |L^d|$. This yields a triangulation K_i^d of S^d and a simplicial map $g_i^{d+1}\colon K_i^d \to L^d$ homotopic to ϕ_i^{d+1}.
- From K_i^d, we construct a simplicial ball $B(K_i^d)$ with boundary K_i^d and glue it to L^d along this boundary via g_i^{d+1}. This amounts to the simplicial mapping cone of g_i^{d+1}.

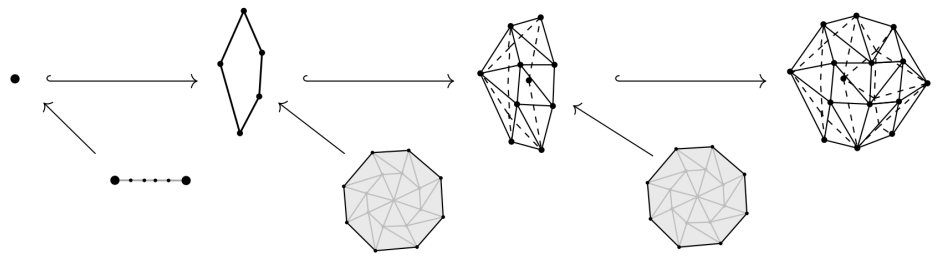
After all $(d+1)$-cells have been attached, the resulting complex L^{d+1} is homotopy equivalent to the skeleton X^{d+1}, and the procedure can be iterated in the next dimension.



(a) Diagram induced by the CW structure on S^2 in Figure 1a.



(b) Geometric visualization of the diagram above.



(c) Simplicial approximation of the diagram.

Figure 2 Schematic view of the simplicial approximation procedure for CW complexes.

Computational obstacles. Conventionally, the simplicial approximation of $\phi_i^{d+1}\colon S^d \to |L^d|$ is obtained as follows: one starts from any triangulation K of S^d and repeatedly applies barycentric subdivision until the triangulation is fine enough. More precisely, one stops when the map satisfies the *star condition*: each closed star of a vertex $v \in K$ is mapped by ϕ_i^{d+1} into the open star of some vertex $w \in L^d$. The simplicial approximation theorem ensures that this procedure terminates [72, Theorem 2C.1]. The assignment of vertices $v \mapsto w$ defines a simplicial map g_i^{d+1} homotopic to ϕ_i^{d+1} (through a homotopy that is linear on each simplex).

In practice, this approach quickly becomes intractable for two reasons. First, barycentric subdivision introduces a large number of vertices: a d-simplex is converted into a complex with $2^{d+1}-1$ vertices. In Section 2, we propose to use instead the *Delaunay complexes* and their refinements. We show in Theorem 4 that Delaunay refinement enjoys better approximation properties than barycentric subdivision: the simplices shrink more rapidly.

Secondly, the star condition itself is too coarse: each facet of L^d requires $d+1$ vertices of K that are mapped into it. This contributes further to the exponential growth in the number of vertices. In Section 4, we avoid the star condition altogether by decomposing the problem into two parts: a geometric step (constructing a homotopy equivalence) and a combinatorial step (constructing a simplicial map). Proposition 12 ensures that this procedure is correct.

Last, the standard construction of a simplicial mapping cone, going back to Cohen [47], relies on the barycentric subdivision of K, which again leads to large complexes. In Section 3, we consider a lighter triangulation, based on *staircase triangulation* of products. Proposition 7 shows that this object is homotopy equivalent to the standard mapping cone.

2 Successive refinements of spherical Delaunay triangulations

Our construction begins with a refinement scheme for spherical Delaunay complexes, which generates arbitrarily fine triangulations and is well suited to concrete implementation.

2.1 Spherical Delaunay triangulations

Definition

Let $X \subset \mathbb{R}^d$ be finite. A subset of $d+1$ points has the *empty circle property* if its circumscribing (open) ball is empty of points of X. These subsets form the facets of an abstract simplicial complex $\mathrm{Del}(X)$, called the *Delaunay complex*. Under the genericity assumption that no subset of $d+2$ points lies on the same sphere, $\mathrm{Del}(X)$ is naturally embedded in $\mathbb{R}^d$ [20].

These definitions adapt to the spherical case: given a subset X of the unit sphere $S^d \subset \mathbb{R}^{d+1}$, the empty circle property is understood with respect to the geodesic distance on S^d; the resulting complex is called the *spherical Delaunay complex* and is still denoted $\mathrm{Del}(X)$. If no subset of $d+2$ points lies on the same geodesic sphere, then it is naturally embedded in $\mathbb{R}^{d+1}$. We shall implicitly make this assumption throughout the article.

It is well-known that the spherical Delaunay complex coincides with the boundary of the convex hull of its vertices, thus reducing the computation of $\mathrm{Del}(X)$ to $\mathrm{conv}(X)$ [31]. In our implementation, we used `Qhull`, a popular software package for computing convex hulls [10].

Point location

A natural map $|\mathrm{Del}(X)| \to S^d$ is given by the scaling $x \mapsto x/\|x\|$, which is a homeomorphism provided that the origin is included in the interior of the convex hull of X. A set X satisfying this assumption will be referred to as *admissible*, and $\mathrm{Del}(X)$ as an *admissible triangulation*.

We call the inverse homeomorphism $r\colon S^d \to |\mathrm{Del}(X)|$ the *radial projection*. Computationally speaking, the radial projection of $x \in S^d$ is found by identifying a facet $\sigma \in \mathrm{Del}(X)$ to which $r(x)$ belongs, then computing a simple line/hyperplane intersection.

We perform point location via a conventional *Jump-and-Walk* strategy [93]: it consists of choosing a first candidate $\sigma \in \mathrm{Del}(X)$, hopefully close to $r(x)$, then walking through its neighbor facets until reaching one that contains $r(x)$. Unlike popular software packages such as `STRIPACK` or `CGAL` [109, 98], our implementation does not make use of spherical geometric predicates. Instead, we project $\mathrm{Del}(X)$ into the tangent space of S^d at x via the stereographic projection $p\colon S^d \setminus \{-x\} \to T_xS^d$ and work in the induced Euclidean triangulation (the simplices are replaced with the convex hull of their vertices). This approach lets us reuse Euclidean routines already present in our code. In this step, we exclude from $\mathrm{Del}(X)$ all simplices incident to $-x$, since their images under p are not defined.

A caveat is that stereographic projection distorts geodesics: in general, the image geodesic simplex $p(|\sigma|)$ need not coincide with the linear simplex $\mathrm{conv}(\{p(v_0), \ldots, p(v_d)\})$ on the image of its vertices. Consequently, a point might map to the "wrong" linear simplex. The next lemma shows that this cannot happen at the pole x, which justifies our procedure.

▶ **Lemma 1.** *Let $\sigma = [v_0, \ldots, v_d] \in \mathrm{Del}(X)$ and $x \in S^d$ such that $r(x) \in |\sigma|$. Then $p(x) \in \mathrm{conv}(\{p(v_0), \ldots, p(v_d)\})$, where p is the stereographic projection at x.*

2.2 Global Delaunay refinements

Given a spherical Delaunay triangulation $\mathrm{Del}(X_0)$ with vertex set $X_0 \subset S^d$, one obtains a finer triangulation by choosing additional points $Y_0 \subset S^d$ and building the Delaunay complex on $X_1 = X_0 \cup Y_0$. The new points are called *Steiner points*, and this procedure is known as a *Delaunay refinement*. This construction can be iterated, yielding a sequence of Steiner points Y_0, Y_1, Y_2, ... and complexes $\mathrm{Del}(X_0)$, $\mathrm{Del}(X_1)$, $\mathrm{Del}(X_2)$, ...

In computational geometry, Delaunay refinement lies at the core of popular algorithms for generating and refining Euclidean meshes; see [43] for a modern presentation. For instance, Chew's and Ruppert's algorithms [44, 45, 99] take a set of input constraints (e.g., boundary segments that must appear as edges) and iteratively improve the mesh by eliminating poor-quality triangles. The refinement step consists of marking a bad triangle and inserting its circumcenter as a new vertex, after which the Delaunay triangulation is recomputed.

Our objective here is different: we seek triangulations whose simplices can be made arbitrarily small, enabling the simplicial approximation of maps described in Section 1.4. To this end, we introduce several families of Delaunay-based refinements, visualized in Figure 3.

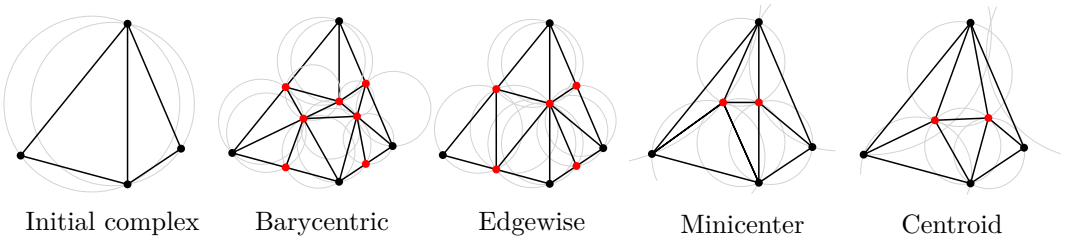
Barycentric refinement: Inspired by barycentric subdivision, we let the Steiner points Y_i be the barycenters of all simplices of $\mathrm{Del}(X_i)$, except its vertices.

Edgewise refinement: We may instead insert the midpoints of all edges of $\mathrm{Del}(X_i)$, in analogy with the popular Coxeter-Freudenthal-Kuhn subdivisions of simplices, also called edgewise subdivisions [17, 54, 67, 97, 82, 80, 46, 27, 32, 19, 28, 29].

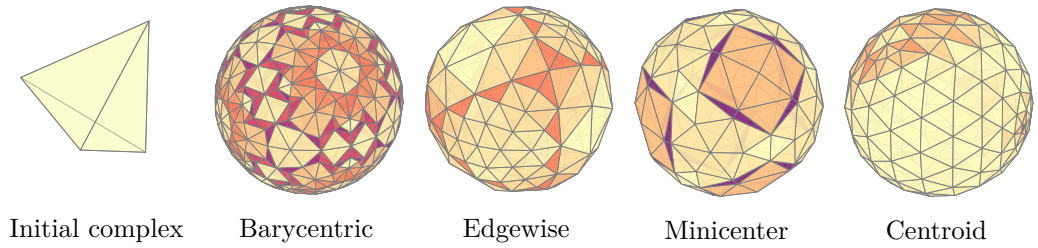
Minicenter refinement: Closer in spirit to conventional Delaunay refinement, we also consider inserting the minicenters of the facets (centers of minimal enclosing balls). We favor minicenters over circumcenters because, unlike circumcenters, they always lie inside the simplex that defines them, which better reflects the local nature of classical subdivision.

Centroid refinement: Finally, we may insert the barycenters of only the *facets* of $\mathrm{Del}(X_i)$ (the maximal simplices), standing as a cheaper alternative to barycentric refinement.

In all cases, the *spherical* Steiner points are considered (i.e., spherical barycenters or minicenters). This is equivalent to computing their Euclidean counterpart in the ambient space $\mathbb{R}^{d+1}$ and projecting them onto the sphere; see, for instance, Fiedler's book [63, Section 5.4].



(a) Local picture: A Delaunay complex on a set of four points in $\mathbb{R}^2$ and the four refinements proposed.



(b) Global picture: Spherical Delaunay complexes after three or four refinements of an initial triangulation of S^2 (the boundary of the standard simplex). The colors indicate the simplex ratio inradius/circumradius.

Figure 3 We employ Delaunay refinement as a subdivision scheme for triangulations of S^d.

Note that, strictly speaking, $\mathrm{Del}(X_{i+1})$ is not a subdivision of $\mathrm{Del}(X_i)$: when realized on the sphere, a simplex of $\mathrm{Del}(X_{i+1})$ need not be contained in a single simplex of $\mathrm{Del}(X_i)$. This can be seen in Figure 3a, where only the edgewise and minicenter refinements are subdivisions. As a consequence, it is not immediate that simplices become smaller under refinement. In particular, adding a point to a Delaunay triangulation can increase the maximal edge length, as illustrated in Figure 4 (shown in the plane for clarity). Instead, we show that the *maximal (spherical) circumradius* of $\mathrm{Del}(X_i)$, denoted $\rho_{\mathrm{circ}}(\mathrm{Del}(X_i))$, decreases monotonically to zero. Since the maximal diameter of the simplices of $\mathrm{Del}(X_i)$ is at most twice its maximal circumradius, it follows that the maximal diameter also tends to zero.

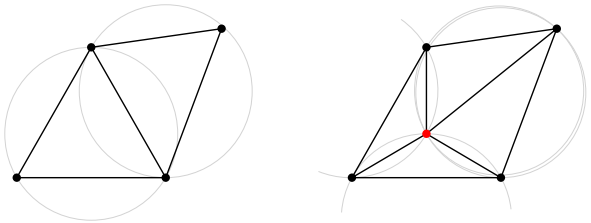


Figure 4 Adding a point to a Delaunay complex may increase the maximal edge length.

A convenient quantity to study a Delaunay complex on a subset $X \subset S^d$ is its *covering radius* (also known as sampling radius) [21, 20]. On the sphere, it is defined by

$$\rho_{\mathrm{cov}}(X) = \sup_{y \in S^d} \inf_{x \in X} \mathrm{d}(x, y),$$

where $\mathrm{d}(x, y)$ is the geodesic (great-circle) distance on the sphere. We note that the set X is admissible as long as $\rho_{\mathrm{cov}}(X) < \pi/2$ (i.e., X is not contained in a closed hemisphere).

The maximal circumradius and covering radius are related by the following standard result.

▶ **Lemma 2.** *For every admissible finite subset $X \subset S^d$, it holds that $\rho_{\mathrm{circ}}(\mathrm{Del}(X)) = \rho_{\mathrm{cov}}(X)$.*

We take advantage of this shift to the covering radius to prove the following lemma.

▶ **Lemma 3.** *Consider a finite subset $X \subset S^d$ and let Y denote the Steiner points associated with* $\mathrm{Del}(X)$*. Assume that $\rho_{\mathrm{cov}}(X) \leq \pi/3$. Then*

$$\rho_{\mathrm{cov}}(X \cup Y) \leq \alpha \rho_{\mathrm{cov}}(X),$$

where $\alpha = \alpha' / \cos(\rho_{\mathrm{cov}}(X))$, and α' depends on the chosen refinement, as given in the table

Refinement	Edgewise	Minicenter	Centroid
α'	$1/\sqrt{2}$	$1/\sqrt{2}$	$d/(d+1)$

The quantity $\cos(\rho_{\mathrm{cov}}(X))$ reflects the spherical distortion of lengths; it goes to 1 as the covering radius goes to zero. A similar lemma can be obtained in the Euclidean case, without this factor. We point out that we have not been able to obtain a satisfactory bound for barycentric refinement; for lack of a better estimate, we use the bound for edgewise refinement.

By iterating this lemma, we obtain our main result on Delaunay refinement.

▶ **Theorem 4.** *Assume the initial sample $X_0 \subset S^d$ is sufficiently dense so that $\alpha < 1$, where $\alpha = \alpha(X_0)$ is defined in Lemma 3. Then the n^{th} iteration of Delaunay refinement satisfies*

$$\rho_{\mathrm{cov}}(X_n) \leq \alpha^n \rho_{\mathrm{cov}}(X_0).$$

In particular, the maximal diameter of simplices of $\mathrm{Del}(X_n)$ *goes to zero.*

▶ Remark 5. This result highlights a notable distinction between Delaunay refinement and standard subdivision. While barycentric refinement reduces diameters asymptotically by $1/\sqrt{2}$ (at most), the best bound for barycentric subdivision is $d/(d+1)$. Likewise, centroid refinement reduces them by $d/(d+1)$, even though it introduces only one vertex per facet.

3 Simplicial mapping cones with staircase triangulations

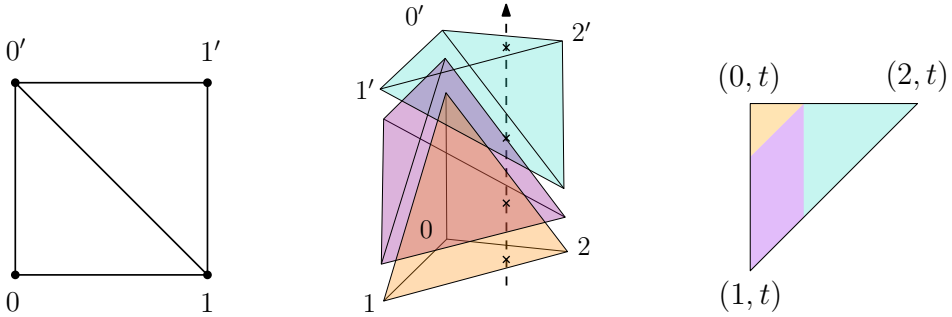
In this section, we assume that a simplicial approximation $g\colon K \to L$ of $f\colon S^d \to |L|$ is given. We build a simplicial mapping cone $C_{\mathrm{simp}}(g)$ and show it is homotopy equivalent to the usual mapping cone $C(f)$. The construction proceeds by building a simplicial ball $B(K)$ through staircase triangulation, which we first recall. Although this construction already appears in the literature, notably in [105, Exercise E, p. 151] and [12, Proposition 2], we include a detailed account here in order to collect the properties that will be used in the sequel.

3.1 Triangulation of the ball

Staircase triangulation of the prism

To triangulate the Cartesian product of a d-simplex σ with an interval $I = [0, 1]$, the *staircase triangulation* consists of first ordering the vertices $\{v_0, \ldots, v_d\}$ of σ, taking a copy $\{v'_0, \ldots, v'_d\}$, and inserting the simplices $\sigma_k = [v_k, \ldots, v_d, v'_0, \ldots, v'_k]$ for all $k \in \{0, \ldots, d\}$ [86]. We shall refer to the product $|\sigma| \times I$ as a *prism*, and its face $[v_k, \ldots, v_d]$ (resp. $[v'_0, \ldots, v'_k]$) as the *inner face* (resp. the *outer face*). Geometrically, they correspond to $|\sigma| \times \{0\}$ and $|\sigma| \times \{1\} \subset |\sigma| \times I$.

The relation $\sigma_k < \sigma_{k+1}$ defines an order on the prism's $d+1$ facets. In particular, the following observation will be useful later: *vertical* straight lines in the prism – i.e., of the form $t \mapsto (x, t)$ for a certain $x \in |\sigma|$ – cross each facet consecutively. Indeed, for $k \in \{1, \ldots, d-1\}$, the only neighboring facets of σ_k are σ_{k-1} and σ_{k+1}; see Figures 5a and 5b. On the other hand, *horizontal* sections of the prism – i.e., sets $|\sigma| \times \{t\}$ for $t \in [0, 1]$ – inherit subdivisions in polyhedral cells; see Figure 5c. These are closely related to *mixed subdivisions* [86, 100].



(a) The square $|\Delta^1| \times I$ is triangulated with two triangles. **(b)** The prism $|\Delta^2| \times I$ is triangulated with three tetrahedra. **(c)** Horizontal sections inherit a subdivision into polygonal cells.

Figure 5 A triangulation of the prism $|\sigma| \times I$ is obtained by ordering the vertices $\{v_0, \dots, v_d\}$ of σ, taking a copy $\{v'_0, \dots, v'_d\}$, and inserting the simplices $\sigma_k = [v_k, \dots, v_d, v'_0, \dots, v'_k]$ for $0 \le k \le d$.

Filling the sphere

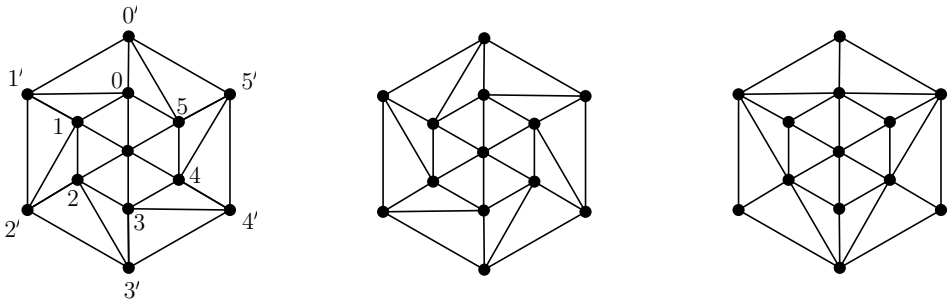
Let K be an admissible triangulation of the sphere $S^d \subset \mathbb{R}^{d+1}$ (a convex hull of unit vectors that contains the origin). We obtain a triangulation of the unit ball B^{d+1} in three steps.

- The polyhedron $|K|$ is embedded in $\mathbb{R}^{d+1}$ and called the *outer layer*. We prime its vertex labels ($0'$, $1'$, *etc.*). Besides, a copy of $|K|$ is taken and scaled by a factor $\rho_{\text{inner}} \in (0, 1)$; it is seen as a triangulation of the sphere of radius ρ_{inner}, referred to as the *inner layer*.
- Each simplex of the outer layer is connected to the corresponding inner-layer simplex via staircase triangulation, forming a prism. Together, these prisms yield a triangulation of the spherical shell of radii $(\rho_{\text{inner}}, 1)$, which we call the *outer shell*.
- Last, the origin is added to the triangulation as a new vertex, over which the inner layer is coned, forming the *inner ball*.

The resulting geometric simplicial complex, denoted $B(K)$, is a triangulation of the unit ball.

In our implementation, we chose $\rho_{\text{inner}} = 1/2$. Besides, because the construction uses staircase triangulations, it depends on a choice of ordering of the vertices in each facet of K. These orderings must be compatible across neighboring facets. Equivalently, the construction requires an "orientation" on K, by which we mean an orientation of its edges such that no facet contains a directed cycle. Different choices may lead to different $B(K)$; see Figure 6.

Each facet $\sigma \in K$ generates a *sector*, defined as the subcomplex $\text{Sect}(\sigma) \subset B(K)$ containing the origin, its cone with σ seen in the inner layer, and the prism built on it. Most of the constructions to follow will be carried out sector by sector.



(a) $0 \to 1 \to 2 \to 3 \to 4 \to 5 \to 0$. **(b)** $5 \to 4 \to 3 \to 2 \to 1 \to 0 \to 5$. **(c)** $0 \to 1 \leftarrow 2 \to 3 \leftarrow 4 \to 5 \leftarrow 0$.

Figure 6 The simplicial ball $B(K)$ built from K depends on an orientation of the edges of K.

Radial normalization

A number of natural homeomorphisms $|B(K)| \to B^{d+1}$ exist. For instance, one could lift the vertices of $B(K)$ to the upper $(d+1)$-hemisphere of $\mathbb{R}^{d+2} \supset \mathbb{R}^{d+1} \times \{0\}$ via orthographic or stereographic projection, build geodesic simplices, and take them back to $B^{d+1} \subset \mathbb{R}^{d+1}$. However, we found that the idea of *radial normalization* was more appropriate for our problem.

As for any convex domain, the *gauge function* of $|B(K)|$ is defined for all $x \in \mathbb{R}^{d+1}$ as

$$J(x) = \inf\{t > 0 \mid t^{-1}x \in |B(K)|\}.$$

It is a convex, positively homogeneous function. The reciprocal of the gauge is known as the *radial function*, used in the study of star-shaped sets [71]. It can be written as

$$J(x)^{-1} = \sup\{t > 0 \mid tx \in |B(K)|\}.$$

If x is a unit vector, then $J(x)^{-1}$ is the length of the part of $[0, x]$ contained in $|B(K)|$. In particular, the minimum of $J(x)^{-1}$ over the unit sphere is equal to the polyhedron's inradius.

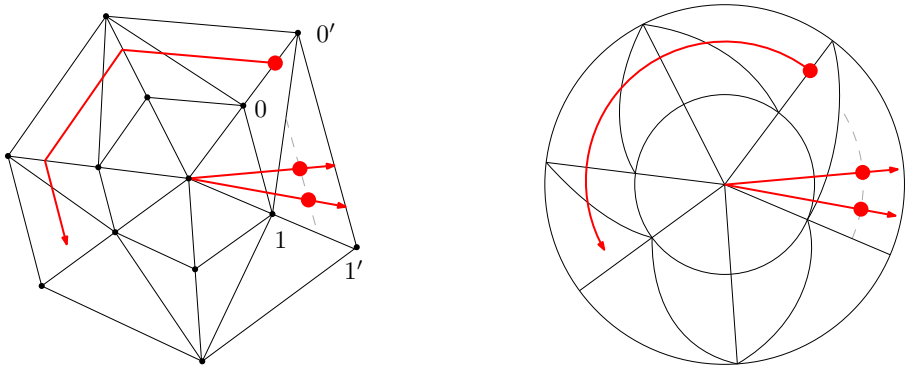
Our preferred homeomorphism $\nu\colon |B(K)| \to B^{d+1}$ is the *radial normalization*, defined as

$$\nu(x) = J\left(\frac{x}{\|x\|}\right)x,$$

with inverse $\nu^{-1}(x) = J(x/\|x\|)^{-1}x$. Although not explicitly written, ν depends on $|B(K)|$. One of its main advantages is its simple geometric behavior, illustrated in Figure 7.

▶ **Lemma 6.** *Under inverse radial normalization* $\nu^{-1}\colon B^{d+1} \to |B(K)|$,

- *rays through the origin are mapped to rays through the origin;*
- *circular arcs – i.e., intersections of linear planes with spheres centered at the origin – are mapped to paths which are linear in each sector* $|\mathrm{Sect}(\sigma)| \subset |B(K)|$ *and parallel to* $|\sigma|$.



(a) The polyhedron $B(K)$ seen in $\mathbb{R}^{d+1}$. **(b)** Its image, the Euclidean ball $B^{d+1} \subset \mathbb{R}^{d+1}$.

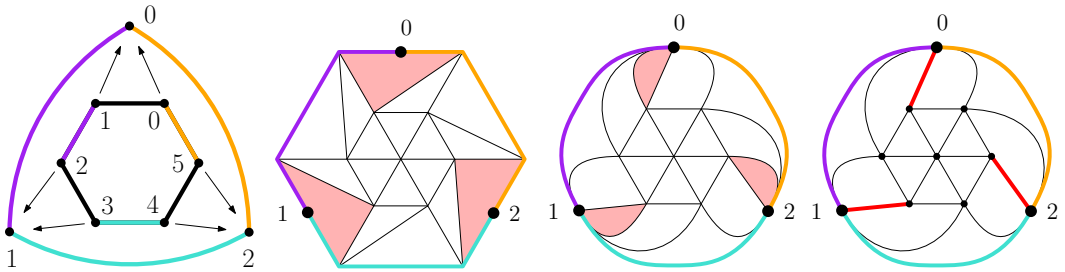
Figure 7 Radial normalization yields a homeomorphism $\nu\colon |B(K)| \to B^{d+1}$.

3.2 Homotopy equivalence between the mapping cones

We still consider a simplicial approximation $g\colon K \to L$ to $f\colon S^d \to |L|$. In the previous section we built a simplicial ball and a homeomorphism $B^{d+1} \to |B(K)|$. We now face three distinct gluings, represented in Figure 8, which we will show are homotopy equivalent:

$$|L| \cup_f B^{d+1} \longrightarrow |L| \cup_{|g|} B^{d+1} \longrightarrow |L \cup_g B(K)|.$$

The first two are the (standard) mapping cones of f and $|g|$, also denoted $C(f)$ and $C(|g|)$. The latter is the *simplicial mapping cone* of g, also denoted $C_{\text{simp}}(g)$. We define it as the quotient of $B(K) \sqcup L$ by the relation $v \sim g(v)$ for all vertices v in the outer layer $K \subset B(K)$.



(a) $g\colon K \to L$. **(b)** $|L| \cup_f B^{d+1}$. **(c)** $|L| \cup_{|g|} |B(K)|$. **(d)** $L \cup_g B(K)$.

Figure 8 The mapping cones involved in our problem. Here, $f\colon |K| \to |L|$ is the identity between two triangulations of S^1 on 6 and 3 vertices, and g is the simplicial map $0, 1 \mapsto 0$; $2, 3 \mapsto 1$; $4, 5 \mapsto 2$.

First, it is a standard fact that mapping cones built from homotopic maps are homotopy equivalent [72, Proposition 0.18]. More precisely, since our domains are balls, an explicit homotopy equivalence $|L| \cup_f B^{d+1} \to |L| \cup_{|g|} B^{d+1}$ is given by

$$x \longmapsto \begin{cases} x & \text{if } \ x \in |L|, \\ x/\rho_{\text{hom}} & \text{if } \ x \in B^{d+1} \ \text{ and } \ \|x\| < \rho_{\text{hom}}, \\ H\big(x/\|x\|,\ (\|x\| - \rho_{\text{hom}})/(1-\rho_{\text{hom}})\big) & \text{if } \ x \in B^{d+1} \ \text{ and } \ \|x\| \geq \rho_{\text{hom}}. \end{cases} \tag{1}$$

where H is a homotopy between $|g|$ and f, and $\rho_{\text{hom}} \in (0,1)$ is a parameter, chosen as 0.9 in our implementation. Visually, the homotopy is performed by scaling the ball B^{d+1} by a factor $1/\rho_{\text{hom}}$ and by using the outer shell $\{x \mid \|x\| \geq \rho_{\text{hom}}\}$ to interpolate between $|g|$ and f.

Second, to compare the remaining two gluings, we can simply use the quotient map

$$q\colon |L| \cup_{|g|} |B(K)| \to |L \cup_g B(K)|.$$

It has the effect of collapsing simplices on which g is not injective; compare Figures 8c and 8d.

▶ **Proposition 7.** *The quotient map q is a homotopy equivalence.*

We close this section with a result that will help us navigate the mapping cone.

▶ **Lemma 8.** *Let $x \in |B(K)|$. Under the projection $p\colon |B(K)| \to |L \cup_g B(K)|$, the image of the partial ray $\{tx \mid 0 \leq t \leq 1/\|x\|\}$ only depends on the image of x.*

4 Simplicial approximation as a list homomorphism problem

The final ingredient in our construction is a more efficient simplicial approximation. We consider a continuous map $f\colon S^d \to |L|$ and a triangulation K of S^d. For each vertex v of K, the point $f(v)$ lies in the geometric realization of a unique simplex of L, called its *carrier* and denoted $\text{carr}(f(v))$. We seek a simplicial map $g\colon K \to L$ such that $g(v) \in \text{carr}(f(v))$ for all vertices. The existence of such a map is a purely combinatorial question that we address as a constraint satisfaction problem. A further issue is the existence of a homotopy H between f and $|g|$; we formulate the problem in the framework of locally equiconnected spaces.

4.1 Constructing the homotopy

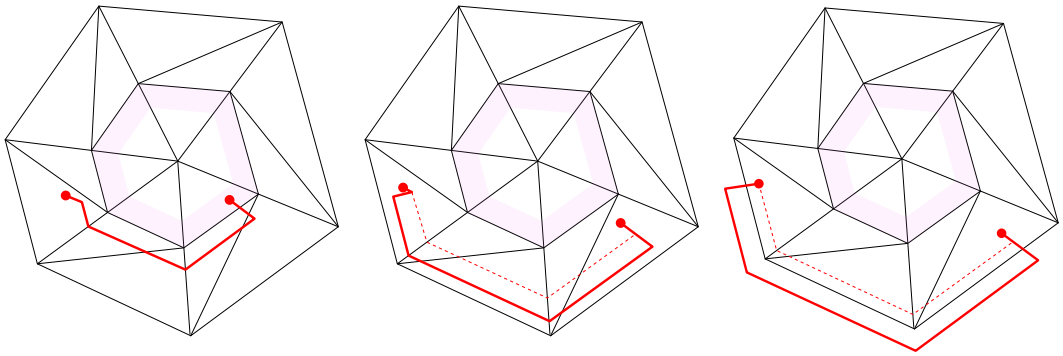
A practical framework for explicitly constructing a homotopy is provided by the theory of *locally equiconnected spaces* (LEC) introduced by Dugundji in 1965 [51]. Namely, Y is LEC if there exists a neighborhood $U \subset Y \times Y$ of the diagonal and a continuous map $\Pi\colon U \times I \to Y$ such that $\Pi(x, y, 0) = x$, $\Pi(x, y, 1) = y$ and $\Pi(x, x, t) = x$ for all $(x, y, t) \in U \times I$. The map Π is called an *equiconnecting map*, and the pair (U, Π) is called *LEC-data*.

In this section, we aim to build LEC-data for simplicial mapping cones $Y = C_{\text{simp}}(g)$ on simplicial maps $g\colon K \to L$. Dyer and Eilenberg [53] have shown how to build LEC-data of *standard* mapping cones $C(|g|)$, provided that both $|K|$ and $|L|$ are LEC. Their construction, however, does not descend to the quotient $C_{\text{simp}}(g)$. We present here a closely related construction, adapted to simplicial mapping cones, in the particular case where K is a sphere.

In general, it is too much to expect an equiconnecting map on Y: we are able to build one only when $g\colon K \to L$ is *2-distance injective*, i.e., injective when restricted to the closed star of each vertex. In the language of graphs, this is equivalent to saying that g is a 2-distance coloring of the 1-skeleton of K [81, 30]. Without this hypothesis, we instead construct a *local motion planner* [60], i.e., a continuous map $\Pi\colon U \times I \to Y$, where $U \subset Y \times Y$ is a neighborhood of the diagonal, satisfying $\Pi(x, y, 0) = x$ and $\Pi(x, y, 1) = y$ for all $(x, y) \in U$. In contrast with equiconnecting maps, the path $t \mapsto \Pi(x, x, t)$ need not be constant.

▶ **Theorem 9.** *If $g\colon K \to L$ is 2-distance injective and $|L|$ is endowed with LEC-data, then $|C_{\text{simp}}(g)|$ also admits LEC-data (U, Π). When g is not 2-distance injective, the same result holds for local motion planners instead of equiconnecting maps.*

As illustrated in Figure 9, we construct a planner on $C_{\text{simp}}(g)$ by first defining it on $B(K)$, through a combination of "elementary paths" (rays, straight paths, circular arcs). We prove that they descend along the projection map $|B(K)| \to |C_{\text{simp}}(g)|$, yielding a well-defined local planner that can subsequently be spliced with the original planner on $|L|$.



(a) Points are connected through a combination of elementary paths.

(b) For distant points, the paths are pushed to the boundary.

(c) Close to the boundary, paths are interpolated with those on $|L|$.

Figure 9 We build a local motion planner on $C_{\text{simp}}(g) = L \cup_g B(K)$ by first defining it on $B(K)$.

▶ Note 10. Equiconnecting maps or planners allow us to test the homotopy between maps. Indeed, two maps $a, b\colon X \to Y$ are homotopic whenever $(a(x), b(x)) \in U$ for all $x \in X$; a homotopy is given by $H(x, t) = \Pi(a(x), b(x), t)$. We say that the maps are *U-close*. During the proof of Theorem 9, we describe U explicitly. In particular, for points x, y in the ball $|B(K)|$ and their images in the mapping cone via $|B(K)| \to |C_{\text{simp}}(g)|$, the corresponding pair lies in U when x and y are sufficiently close to the origin or when they are not antipodal.

4.2 Simplicial approximation routine

Planner condition

We return to the problem of simplicial approximation for $f\colon S^d \to |L|$ where L is a mapping cone $L = L_0 \cup_{g_0} B(K_0)$ now endowed with a local motion planner (U, Π). Given a triangulation K of S^d, we ask whether it is fine enough so that the planner can be applied on each facet.

More precisely, for each facet $\sigma = [v_0, \dots, v_d] \in K$ we look for a simplicial assignment

$$v_i \mapsto g(v_i) \in \operatorname{carr}(f(v_i))$$

such that, on $|\sigma|$, the continuous map f and the linear map $|g|$ are U-close, i.e., $(f(x), |g|(x)) \in U$ for all $x \in |\sigma|$. Let us assume for simplicity that $f(|\sigma|)$ is contained in the last cell $B(K_0)$ of $L = L_0 \cup_{g_0} B(K_0)$; the general case is treated recursively along the filtration.

As explained in Note 10, a pair $(x, y) \in B(K_0) \times B(K_0)$ belongs to U provided the points are sufficiently close to the origin or are not antipodal ($x/\|x\| \neq -y/\|y\|$). To guarantee this condition uniformly on $f(|\sigma|)$, we estimate its size using the edge lengths of $|\sigma|$ and a Lipschitz bound λ for f. This Lipschitz constant is computed explicitly for each f.

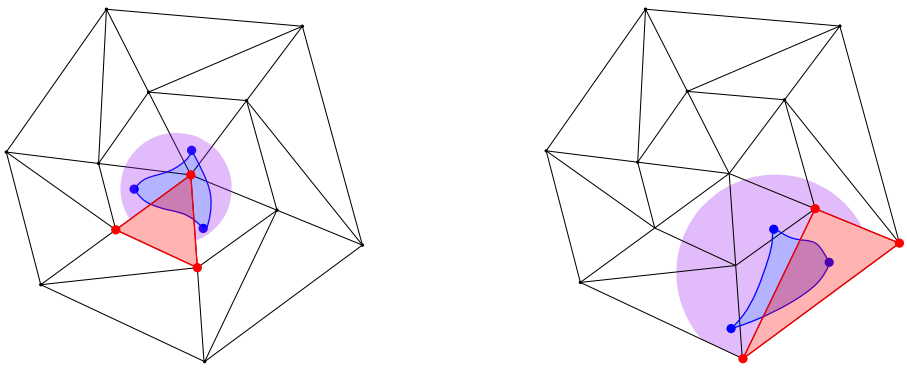
▶ **Lemma 11.** *Let $\sigma = [v_0, \dots, v_d]$ be a geometric simplex in $\mathbb{R}^n$ and $f\colon |\sigma| \to \mathbb{R}^m$ a λ-Lipschitz continuous map. For $I \subset \{0, \dots, d\}$ nonempty, define the barycenter and length*

$$c_I = \frac{1}{|I|} \sum_{i \in I} f(v_i) \quad \text{and} \quad r_I = \max_{0 \le k \le d} \frac{1}{|I|} \sum_{i \in I} \|v_k - v_i\|.$$

Then $f(|\sigma|)$ is included in the closed ball $B(c_I, \lambda r_I)$.

In our implementation we use the case $I = \{0, \dots, d\}$ only, corresponding to the barycenter of all vertices $f(v_i)$, although the estimate can be sharpened by considering other subsets I.

If σ is small enough, Lemma 11 and the description of U ensure that the required assignment g exists: σ is mapped to inner vertices if $f(|\sigma|)$ is close to the origin, or $g(\sigma)$ avoids the origin if $f(|\sigma|)$ is close to the boundary; see Figure 10. We refer to this requirement on σ as the *planner condition*. If the condition fails on some facet, we subdivide K until it is satisfied. The termination of this procedure is established in Proposition 12.



(a) Image $f(|\sigma|)$ close to the origin. **(b)** Image $f(|\sigma|)$ away from the origin.

Figure 10 For every facet $\sigma \in K$, the *planner condition* checks whether the image $f(|\sigma|)$ (in blue) is U-close to a simplex τ of L (in red). Lemma 11 allows us to enclose $f(|\sigma|)$ in a ball (in purple).

List homomorphism problem

Once the planner condition is satisfied on K, we obtain, for each facet $\sigma \in K$, a collection of admissible simplices $\tau_1, \tau_2, \ldots$ of L. From these, we derive for each vertex $v \in V(K)$ a set $\mathcal{L}(v) \subset V(L)$ of admissible images, thus defining a map to the power set

$$\mathcal{L}\colon V(K) \to \mathcal{P}(V(L))$$

Given these lists of candidates, the remaining task is to find a simplicial map g with $g(v) \in \mathcal{L}(v)$ for every vertex. The motion planner ensures that such a map is homotopic to f.

In this form, the problem is essentially an instance of the *list homomorphism problem*, denoted `LHom`. Conventionally, it is formulated for graphs rather than simplicial complexes, and one typically forbids collapsing an edge to a vertex. It is known that `LHom` can be solved in polynomial time when L is a bi-arc graph and is NP-complete otherwise [62, 55].

In practice, we treat our instance as a constraint satisfaction problem and solve it using the `CP-SAT` solver from the `OR-Tools` suite [95, 96]. If the resulting problem is unsatisfiable, we refine the triangulation K and repeat the procedure, as formalized in Algorithm 1.

Delaunay simplifications

Once a simplicial map $g\colon K \to L$ has been found, we apply a postprocessing step that we call *Delaunay simplification*. Since the simplicial sphere K arises as a Delaunay complex $K = \mathrm{Del}(X)$ on a finite set $X \subset S^d$, we seek a subset $X' \subset X$ such that the induced vertex map $g'\colon \mathrm{Del}(X') \to L$ is still simplicial and homotopic to f.

Concretely, we inspect the vertices of X one by one, remove a candidate vertex, recompute the Delaunay complex, and then check whether the new facets still satisfy the planner condition and whether the induced map remains simplicial. This procedure is efficient, since deleting a point $v \in X$ only affects the open star of v in $\mathrm{Del}(X)$. Besides, to promote larger simplices, we preferentially remove vertices that belong to facets with the smallest inradius.

Local Delaunay refinements

A drawback of our method is that it refines the entire complex, even when the obstruction is localized on a few facets. To address this, we introduce a *local refinement* strategy. If the solver reports that the instance is infeasible, we instead consider a weaker problem that we call `LHomDrop`: determine a minimal set of facets $\sigma_0, \ldots, \sigma_k$ of K whose removal makes the instance feasible. We then perform a local refinement by inserting Steiner points (see Section 2.2) only on the facets blamed by the solver. In practice, we impose a time limit and use the best solution found by the solver so far (30 seconds in our implementation).

To accelerate the procedure, we apply `LHom` iteratively on the i-skeleta and use the solution in dimension i as a hint for dimension $i+1$. If the instance is infeasible in dimension i, we call `LHomDrop` to identify which i-simplices are to blame, and refine their maximal cofaces.

In the same spirit, when enforcing the planner condition we may refine only those facets on which the condition fails. We write down our procedure, both for global and local refinements, in Algorithm 1 below. We are currently able to prove termination only for the variant with global refinement. However, in all our experiments the local refinement strategy also terminated and produced significantly smaller triangulations.

▶ **Proposition 12.** *With global refinement, Algorithm 1 terminates and produces a simplicial map homotopic to the input continuous map. With local refinement, if it terminates, then the output is also homotopic to the given map.*

Algorithm 1 Simplicial approximation with global or local refinement and simplification.

Input: map $f\colon S^d \to |L|$, triangulation K of S^d, planner (U, Π) on L, scalar $\lambda > 0$
Output: refinement of K and simplicial map $g\colon K \to L$ homotopic to f

repeat
  compute admissible lists $\mathcal{L}(v) \subset V(L)$ via the planner condition;
  solve `LHom` with a SAT solver to obtain g;
  if *no solution* **then**
    apply Delaunay refinement on K (global, or only facets blamed by `LHomDrop`);
until *a solution g is found*;
apply Delaunay simplification to (K, g);
return (K, g);

4.3 Full algorithm

We now assemble the ingredients of Sections 2–4 into the full procedure, summarized in Algorithm 2. The input is a finite CW complex X together with explicit attaching maps for its cells and Lipschitz constants. The output is a finite simplicial complex L together with a homotopy equivalence $X \to |L|$ encoded as a point location routine on $|L|$.

Algorithm 2 Simplicial approximation to CW complexes with global or local refinement.

Input: CW complex X of dimension n with cells e_i^d, attaching maps ϕ_i^d, scalars λ_i
Output: simplicial complex L endowed with a homotopy equivalence $X \to |L|$

$L \leftarrow$ discrete simplicial complex on the 0-cells of X;
for $d = 1, \dots, n$ **do**
  foreach *d-cell e_i^d with attaching map $\phi_i^d\colon S^{d-1} \to X^{d-1}$* **do**
    compose ϕ_i^d with $X^{d-1} \to |L|$ to obtain $f_i^d\colon S^{d-1} \to |L|$;
    use Algorithm 1 to obtain $g_i^d\colon K_i \to L$ homotopic to f_i^d (with global or local refinement, and parameter λ_i);
    construct a simplicial ball $B(K_i)$ and glue it to L along K_i via g_i^d, i.e., let $L \leftarrow L \cup_{g_i^d} B(K_i)$;
    endow $|L \cup_{g_i^d} B(K_i)|$ with an equiconnecting map or a local motion planner;
return L;

▶ **Theorem 13.** *With global refinement, Algorithm 2 terminates and produces a simplicial complex homotopy equivalent to the given CW complex. With local refinement, if it terminates, then the output is also homotopy equivalent to the input CW complex.*

References

1 Eddie Aamari and Clément Levrard. Stability and minimax optimality of tangential Delaunay complexes for manifold reconstruction. *Discrete & Computational Geometry*, 59(4):923–971, 2018. doi:10.1007/s00454-017-9962-z.

2 Michał Adamaszek, Henry Adams, and Florian Frick. Metric reconstruction via optimal transport. *SIAM Journal on Applied Algebra and Geometry*, 2(4):597–619, 2018. doi:10.1137/17M1148025.

3 Henry Adams and Joshua Mirth. Metric thickenings of Euclidean submanifolds. *Topology and its Applications*, 254:69–84, 2019. doi:10.1016/j.topol.2018.12.014.

4 Karim Adiprasito, Sergey Avvakumov, and Roman Karasev. A subexponential size triangulation of $\mathbb{R}P^n$. *Combinatorica*, 42(1):1–8, 2022. doi:10.1007/s00493-021-4602-x.

5 Dominique Attali, André Lieutier, and David Salinas. Vietoris–Rips complexes also provide topologically correct reconstructions of sampled shapes. *Computational Geometry*, 46(4):448–465, 2013. 27th Annual Symposium on Computational Geometry (SoCG 2011). doi:10.1016/j.comgeo.2012.02.009.

6 Bhaskar Bagchi and Basudeb Datta. From the icosahedron to natural triangulations of $\mathbb{C}P^2$ and $S^2 \times S^2$. *Discrete & Computational Geometry*, 46(3):542–560, 2011. doi:10.1007/s00454-010-9281-0.

7 Bhaskar Bagchi and Basudeb Datta. A triangulation of $\mathbb{C}P^3$ as symmetric cube of S^2. *Discrete & Computational Geometry*, 48(2):310–329, 2012. doi:10.1007/s00454-012-9436-2.

8 Sonia Balagopalan. On a vertex-minimal triangulation of $\mathbb{R}P^4$. *The Electronic Journal of Combinatorics*, pages P1–52, 2017. doi:10.37236/5956.

9 Rubén Ballester, Ernst Röell, Daniel Bīn Schmid, Mathieu Alain, Sergio Escalera, Carles Casacuberta, and Bastian Rieck. MANTRA: The Manifold Triangulations Assemblage. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL: https://openreview.net/pdf?id=X6y5CC44HM.

10 C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996. doi:10.1145/235815.235821.

11 Ulrich Bauer. Ripser: efficient computation of Vietoris–Rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423, June 2021. doi:10.1007/s41468-021-00071-5.

12 G. Bell and A. Dranishnikov. On asymptotic dimension of groups acting on trees. *Geometriae Dedicata*, 103(1):89–101, February 2004. doi:10.1023/b:geom.0000013843.53884.77.

13 Mark Bell, Tracy Hall, and Saul Schleimer. Twister (computer software), 2014. URL: https://github.com/MarkCBell/twister.

14 Mark Bell, Joel Hass, Joachim Hyam Rubinstein, and Stephan Tillmann. Computing trisections of 4-manifolds. *Proceedings of the National Academy of Sciences*, 115(43):10901–10907, 2018. doi:10.1073/pnas.1717173115.

15 Bruno Benedetti, Crystal Lai, Davide Lofano, and Frank H Lutz. Random simple-homotopy theory. *Journal of Applied and Computational Topology*, 8(3):531–555, 2024. doi:10.1007/s41468-023-00139-4.

16 Bruno Benedetti and Frank H Lutz. Random discrete Morse theory and a new library of triangulations. *Experimental Mathematics*, 23(1):66–94, 2014. doi:10.1080/10586458.2013.865281.

17 Jürgen Bey. Simplicial grid refinement: On Freudenthal's algorithm and the optimal number of congruence classes. *Numerische Mathematik*, 85(1):1–29, 2000. doi:10.1007/s002110050475.

18 Anders Björner and Frank H Lutz. Simplicial manifolds, bistellar flips and a 16-vertex triangulation of the Poincaré homology 3-sphere. *Experimental Mathematics*, 9(2):275–289, 2000. doi:10.1080/10586458.2000.10504652.

19 Jean-Daniel Boissonnat, Florestan Brunck, Hana Dal Poz Kouřimská, Arijit Ghosh, and Mathijs Wintraecken. Simplicial subdivision of simplices of arbitrary dimension in a space of constant non-zero curvature with bounded quality. Preprint, HAL Id: hal-04297370, 2023. URL: https://hal.science/hal-04297370.

20 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and topological inference*, volume 57. Cambridge University Press, 2018. doi:10.1017/9781108297806.

21 Jean-Daniel Boissonnat, Ramsay Dyer, and Arijit Ghosh. Delaunay stability via perturbations. *International Journal of Computational Geometry & Applications*, 24(02):125–152, 2014. doi:10.1142/S021819591450006X.

22 Jean-Daniel Boissonnat, Ramsay Dyer, and Arijit Ghosh. Delaunay triangulation of manifolds. *Foundations of Computational Mathematics*, 18(2):399–431, 2018. doi:10.1007/s10208-017-9344-1.

23 Jean-Daniel Boissonnat and Arijit Ghosh. Manifold reconstruction using tangential Delaunay complexes. In *Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry*, SoCG '10, pages 324–333, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1810959.1811013.

24 Jean-Daniel Boissonnat and Arijit Ghosh. Manifold reconstruction using tangential Delaunay complexes. *Discrete & Computational Geometry*, 51(1):221–267, 2014. doi:10.1007/s00454-013-9557-2.

25 Jean-Daniel Boissonnat, Leonidas J. Guibas, and Steve Y. Oudot. Manifold reconstruction in arbitrary dimensions using witness complexes. In *Proceedings of the Twenty-Third Annual Symposium on Computational Geometry*, SCG '07, pages 194–203, New York, NY, USA, 2007. Association for Computing Machinery. doi:10.1145/1247069.1247106.

26 Jean-Daniel Boissonnat, Leonidas J. Guibas, and Steve Y. Oudot. Manifold reconstruction in arbitrary dimensions using witness complexes. *Discrete & Computational Geometry*, 42(1):37–70, July 2009. doi:10.1007/s00454-009-9175-1.

27 Jean-Daniel Boissonnat, Siargey Kachanovich, and Mathijs Wintraecken. A compact data structure for high dimensional Coxeter–Freudenthal–Kuhn triangulations. Preprint, HAL Id: hal-03006608, 2020. URL: https://hal.science/hal-03006608.

28 Jean-Daniel Boissonnat, Siargey Kachanovich, and Mathijs Wintraecken. Tracing isomanifolds in $\mathbb{R}^d$ in time polynomial in d using Coxeter–Freudenthal–Kuhn triangulations. In Kevin Buchin and Éric Colin de Verdière, editors, *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2021.17.

29 Jean-Daniel Boissonnat, Siargey Kachanovich, and Mathijs Wintraecken. Tracing isomanifolds in $\mathbb{R}^d$ in time polynomial in d using Coxeter–Freudenthal–Kuhn triangulations. *SIAM Journal on Computing*, 52(2):452–486, 2023. doi:10.1137/21M1412918.

30 Boštjan Brešar, Babak Samadi, and Ismael G Yero. Injective coloring of graphs revisited. *Discrete Mathematics*, 346(5):113348, 2023. doi:10.1016/j.disc.2023.113348.

31 K. Q. Brown. *Geometric transforms for fast geometric algorithms*. Ph.D. thesis, Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, 1980. Report CMU-CS-80-101. URL: https://apps.dtic.mil/sti/tr/pdf/ADA081448.pdf.

32 Florestan Brunck. Iterated medial triangle subdivision in surfaces of constant curvature. *Discrete & Computational Geometry*, 70(3):1059–1089, 2023. doi:10.1007/s00454-023-00500-5.

33 Rhuaidi Burke. Practical software for triangulating and simplifying 4-manifolds. *Journal of Computational Geometry*, 16(2):109–144, July 2025. doi:10.20382/jocg.v16i2a4.

34 Rhuaidi Antonio Burke, Benjamin A. Burton, and Jonathan Spreer. Small triangulations of 4-manifolds and the 4-manifold census. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:16, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2025.28.

35 Benjamin A. Burton, Ryan Budney, William Pettersson, et al. Regina: Software for low-dimensional topology, 2021. URL: http://regina-normal.github.io/.

36 Martin Čadek, Marek Krčál, Jiří Matoušek, Francis Sergeraert, Lukáš Vokřínek, and Uli Wagner. Computing all maps into a sphere. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1–10. SIAM, 2012. doi:10.1137/1.9781611973099.1.

37 Martin Čadek, Marek Krčál, Jiří Matoušek, Francis Sergeraert, Lukáš Vokřínek, and Uli Wagner. Computing all maps into a sphere. *Journal of the ACM (JACM)*, 61(3):1–44, 2014. doi:10.1145/2597629.

38 Martin Čadek, Marek Krčál, Jiří Matoušek, Lukáš Vokřínek, and Uli Wagner. Polynomial-time computation of homotopy groups and Postnikov systems in fixed dimension. *SIAM Journal on Computing*, 43(5):1728–1780, 2014. doi:10.1137/120899029.

39 Martin Čadek, Marek Krčál, and Lukáš Vokřínek. Algorithmic solvability of the lifting-extension problem. *Discrete & Computational Geometry*, 57(4):915–965, 2017. doi:10.1007/s00454-016-9855-6.

40 Frédéric Chazal, David Cohen-Steiner, and André Lieutier. A sampling theory for compact sets in Euclidean space. *Discrete & Computational Geometry*, 41(3):461–479, 2009. doi:10.1007/s00454-009-9144-8.

41 Frédéric Chazal and André Lieutier. Stability and computation of topological invariants of solids in $\mathbb{R}^n$. *Discrete & Computational Geometry*, 37(4):601–617, 2007. doi:10.1007/s00454-007-1309-8.

42 Frédéric Chazal and André Lieutier. Smooth manifold reconstruction from noisy and non-uniform approximation with guarantees. *Computational Geometry*, 40(2):156–170, 2008. doi:10.1016/j.comgeo.2007.07.001.

43 Siu-Wing Cheng, Tamal Krishna Dey, and Jonathan Shewchuk. *Delaunay mesh generation*. CRC Press Boca Raton, 2013. doi:10.1201/b12987.

44 L. Paul Chew. Guaranteed-quality triangular meshes. Technical Report TR-89-983, Department of Computer Science, Cornell University, 1989. URL: https://apps.dtic.mil/sti/html/tr/ADA210101/.

45 L Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the ninth annual symposium on Computational geometry*, pages 274–280, 1993. doi:10.1145/160985.161150.

46 Aruni Choudhary, Siargey Kachanovich, and Mathijs Wintraecken. Coxeter triangulations have good quality. *Mathematics in Computer Science*, 14:141–176, 2020. doi:10.1007/s11786-020-00461-5.

47 Marshall M Cohen. Simplicial structures and transverse cellularity. *Annals of Mathematics*, 85(2):218–245, 1967. doi:10.2307/1970440.

48 Marc Culler, Nathan M. Dunfield, Matthias Goerner, and Jeffrey R. Weeks. SnapPy: A computer program for studying the geometry and topology of 3-manifolds, 2021. URL: http://snappy.computop.org.

49 Basudeb Datta and Jonathan Spreer. Simplicial cell decompositions of $\mathbb{CP}^n$. *Discrete & Computational Geometry*, pages 1–24, 2025. doi:10.1007/s00454-025-00801-x.

50 X. Dousson, J. Rubio, F. Sergeraert, and Y. Siret. *The Kenzo program*. Institut Fourier, Grenoble, 1999. URL: http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/.

51 J. Dugundji. Locally equiconnected spaces and absolute neighborhood retracts. *Fundamenta Mathematicae*, 57:187–193, 1965. Publisher: Instytut Matematyczny Polskiej Akademii Nauk. doi:10.4064/fm-57-2-187-193.

52 Nathan M. Dunfield. A census of exceptional Dehn fillings. In *Characters in Low-Dimensional Topology*, volume 760 of *Contemporary Mathematics*, pages 143–155. American Mathematical Society, 2020. doi:10.1090/conm/760/15289.

53 Eldon Dyer and S. Eilenberg. An adjunction theorem for locally equiconnected spaces. *Pacific Journal of Mathematics*, 41(3):669–685, June 1972. doi:10.2140/pjm.1972.41.669.

54 Herbert Edelsbrunner and Daniel R Grayson. Edgewise subdivision of a simplex. *Discrete & Computational Geometry*, 24(4):707–719, 2000. doi:10.1007/s004540010063.

55 László Egri, Andrei Krokhin, Benoit Larose, and Pascal Tesson. The complexity of the list homomorphism problem for graphs. *Theory of Computing Systems*, 51(2):143–178, 2012. doi:10.1007/s00224-011-9333-8.

56 Graham Ellis. *HAP: Homological Algebra Programming, Version 1.70*, 2025. URL: https://gap-packages.github.io/hap.

57 Henrique Ennes and Clément Maria. Hardness of computation of quantum invariants on 3-manifolds with restricted topology. In *33rd Annual European Symposium on Algorithms (ESA 2025)*, volume 351 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:16, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2025.37.

58 Henrique Ennes and Clément Maria. Compressed data structures for Heegaard splittings, 2025. doi:10.48550/arXiv.2507.11406.

59 Ozan B Ericok and Jeremy K Mason. Quotient maps and configuration spaces of hard disks. *Granular Matter*, 24(3):76, 2022. doi:10.1007/s10035-022-01235-5.

60 Michael Farber. Topological complexity of motion planning. *Discrete & Computational Geometry*, 29(2):211–221, 2003. doi:10.1007/s00454-002-0760-9.

61 Brittany Terese Fasy, Rafal Komendarczyk, Sushovan Majhi, and Carola Wenk. On the reconstruction of geodesic subspaces of $\mathbb{R}^n$. *International Journal of Computational Geometry & Applications*, 32(01n02):91–117, 2022. doi:10.1142/S0218195922500066.

62 Tomás Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42(1):61–80, 2003. doi:10.1002/jgt.10073.

63 Miroslav Fiedler. *Matrices and graphs in geometry*. Number 139 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011. doi:10.1017/CBO9780511973611.

64 Marek Filakovský, Peter Franek, Uli Wagner, and Stephan Zhechev. Computing simplicial representatives of homotopy group elements. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 1135–1151, USA, 2018. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611975031.74.

65 Marek Filakovský, Peter Franek, Uli Wagner, and Stephan Zhechev. Computing simplicial representatives of homotopy group elements. *Journal of Applied and Computational Topology*, 2(3):177–231, 2018. doi:10.1007/s41468-018-0021-5.

66 Marek Filakovský and Lukáš Vokřínek. Are two given maps homotopic? An algorithmic viewpoint. *Foundations of Computational Mathematics*, 20(2):311–330, 2020. doi:10.1007/s10208-019-09419-x.

67 EN Gonçalves, RM Palhares, RHC Takahashi, and RC Mesquita. H2 and Hinf ε-guaranteed cost computation of uncertain linear systems. *IET Control Theory & Applications*, 1(1):201–209, 2007. doi:10.1049/iet-cta:20050334.

68 Dejan Govc, Wacław Marzantowicz, and Petar Pavešić. How many simplices are needed to triangulate a Grassmannian? *Topological Methods in Nonlinear Analysis*, 56(2):429–450, 2020. doi:10.12775/TMNA.2020.027.

69 Brian Green, Mimi Tsuruga, and Nicholas A. Scoville. Estimating the discrete Lusternik-Schnirelmann category. *Topological Methods in Nonlinear Analysis*, 45(1):103, March 2015. doi:10.12775/tmna.2015.006.

70 Leonidas J Guibas and Steve Y Oudot. Reconstruction using witness complexes. *Discrete & Computational Geometry*, 40(3):325–356, 2008. doi:10.1007/s00454-008-9094-6.

71 Guillermo Hansen, Irmina Herburt, Horst Martini, and Maria Moszyńska. Starshaped sets. *Aequationes Mathematicae*, 94(6):1001–1092, 2020. doi:10.1007/s00010-020-00720-7.

72 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002. URL: https://pi.math.cornell.edu/~hatcher/AT/AT.pdf.

73 Jean-Claude Hausmann. *On the Vietoris-Rips complexes and a Cohomology Theory for metric spaces*, pages 175–188. Princeton University Press, December 1996. doi:10.1515/9781400882588-013.

74 Alexander He, James Morgan, and Em K. Thompson. An algorithm to construct one-vertex triangulations of Heegaard splittings. *Journal of Computational Geometry*, 16(1):635–693, November 2025. doi:10.20382/jocg.v16i1a18.

75 William Jaco, Hyam Rubinstein, and Stephan Tillmann. Minimal triangulations for an infinite family of lens spaces. *Journal of Topology*, 2(1):157–180, 2009. doi:10.1112/jtopol/jtp004.

76 William Jaco and J. Hyam Rubinstein. Layered-triangulations of 3-manifolds, 2006. arXiv:math/0603601.

77 William Jaco, J Hyam Rubinstein, and Stephan Tillmann. Coverings and minimal triangulations of 3–manifolds. *Algebraic & Geometric Topology*, 11(3):1257–1265, 2011. doi:10.2140/agt.2011.11.1257.

78 Jisu Kim, Jaehyeok Shin, Frédéric Chazal, Alessandro Rinaldo, and Larry Wasserman. Homotopy Reconstruction via the Čech Complex and the Vietoris-Rips Complex. In *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2020.54.

79 Kevin P. Knudson. Approximate triangulations of Grassmann manifolds. *Algorithms*, 13(7):172, 2020. doi:10.3390/a13070172.

80 Sergey Korotov and Michal Křížek. Red refinements of simplices into congruent subsimplices. *Computers & Mathematics with Applications*, 67(12):2199–2204, 2014. doi:10.1016/j.camwa.2014.01.025.

81 Florica Kramer and Horst Kramer. A survey on the distance-colouring of graphs. *Discrete Mathematics*, 308(2-3):422–426, 2008. doi:10.1016/j.disc.2006.11.059.

82 Tim Kröger and Tobias Preusser. Stability of the 8-tetrahedra shortest-interior-edge partitioning method. *Numerische Mathematik*, 109:435–457, 2008. doi:10.1007/s00211-008-0148-8.

83 Wolfgang Kühnel. Minimal triangulations of Kummer varieties. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 57(1):7–20, 1987. doi:10.1007/BF02941594.

84 Wolfgang Kühnel and Thomas F Banchoff. The 9-vertex complex projective plane. *Math. Intelligencer*, 5(3):11–22, 1983. doi:10.1007/BF03026567.

85 Janko Latschev. Vietoris-Rips complexes of metric spaces near a closed Riemannian manifold. *Archiv der Mathematik*, 77(6):522–528, 2001. doi:10.1007/PL00000526.

86 Carl W Lee and Francisco Santos. Subdivisions and triangulations of polytopes. In *Handbook of Discrete and Computational Geometry*, pages 415–447. Chapman and Hall/CRC, 2017. doi:10.1201/9781315119601.

87 Frank H. Lutz. Combinatorial 3-manifolds with 10 vertices. *Beiträge Algebra Geom.*, 49(1):97–106, 2008. URL: https://ftp6.gwdg.de/pub/EMIS/journals/BAG/vol.49/no.1/5.html.

88 Sushovan Majhi. Demystifying Latschev's Theorem: Manifold Reconstruction from Noisy Data. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 73:1–73:16, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.73.

89 Sushovan Majhi. Demystifying Latschev's theorem: manifold reconstruction from noisy data. *Discrete & Computational Geometry*, 74(2):544–568, 2025. doi:10.1007/s00454-024-00655-9.

90 Fedor Manin and Shmuel Weinberger. Algorithmic aspects of immersibility and embeddability. *International Mathematics Research Notices*, 2024(17):12433–12454, 2024. doi:10.1093/imrn/rnae170.

91 Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The GUDHI library: Simplicial complexes and persistent homology. In *International Congress on Mathematical Software*, pages 167–174. Springer, 2014. doi:10.1007/978-3-662-44199-2_28.

92 John Milnor and James D. Stasheff. *Characteristic Classes*. Princeton University Press, December 1974. doi:10.1515/9781400881826.

93 Ernst P Mücke, Isaac Saias, and Binhai Zhu. Fast randomized point location without preprocessing in two- and three-dimensional delaunay triangulations. In *Proceedings of the twelfth annual symposium on Computational geometry*, pages 274–283, 1996. doi:10.1145/237218.237396.

94 Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 39(1):419–441, 2008. doi:10.1007/s00454-008-9053-2.

95 Laurent Perron and Frédéric Didier. *CP-SAT*. Google, February 2025. Version 9.12. URL: https://developers.google.com/optimization/cp/cp_solver/.

96 Laurent Perron and Vincent Furnon. *OR-Tools*. Google, February 2025. Version 9.12. URL: https://developers.google.com/optimization/.

97 Angel Plaza. The eight-tetrahedra longest-edge partition and Kuhn triangulations. *Computers & Mathematics with Applications*, 54(3):427–433, 2007. doi:10.1016/j.camwa.2007.01.023.

98 Robert J Renka. Algorithm 772: STRIPACK: Delaunay triangulation and Voronoi diagram on the surface of a sphere. *ACM Transactions on Mathematical Software (TOMS)*, 23(3):416–434, 1997. doi:10.1145/275323.275329.

99 Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995. doi:10.1006/jagm.1995.1021.

100 Francisco Santos. The Cayley trick and triangulations of products of simplices. In *Integer points in polyhedra—geometry, number theory, algebra, optimization*, volume 374 of *Contemporary Mathematics*, pages 151–177. Amer. Math. Soc., Providence, RI, 2005. doi:10.1090/conm/374/06904.

101 Soumen Sarkar. Some $\mathbb{Z}_3^n$-equivariant triangulations of $\mathbb{CP}^n$, 2014. arXiv:1405.2568.

102 Francis Sergeraert. Triangulations of complex projective spaces. In *Contribuciones científicas en honor de Mirian Andrés Gómez*, pages 507–519. Universidad de La Rioja, 2010. URL: https://dialnet.unirioja.es/descarga/articulo/3217875.pdf.

103 Jatush V Sheth, Varun Sahni, Sergei F Shandarin, and Bangalore Suryanarayana Sathyaprakash. Measuring the geometry and topology of large-scale structure using SURFGEN: methodology and preliminary results. *Monthly Notices of the Royal Astronomical Society*, 343(1):22–46, 2003. doi:10.1046/j.1365-8711.2003.06642.x.

104 Vin de Silva and Gunnar Carlsson. Topological estimation using witness complexes. In Markus Gross, Hanspeter Pfister, Marc Alexa, and Szymon Rusinkiewicz, editors, *SPBG'04 Symposium on Point - Based Graphics 2004*. The Eurographics Association, 2004. doi:10.2312/SPBG/SPBG04/157-166.

105 Edwin H. Spanier. *Algebraic Topology*. Springer New York, 1981. doi:10.1007/978-1-4684-9322-1.

106 Justin Stoecker and Victor Milenkovic. Interactive visualization of 3D configuration spaces. In *Proceedings of the twenty-ninth annual symposium on Computational geometry*, pages 341–342, 2013. doi:10.1145/2462356.2462358.

107 Thom Sulanke and Frank H. Lutz. Isomorphism-free lexicographic enumeration of triangulated surfaces and 3-manifolds. *European J. Combin.*, 30(8):1965–1979, 2009. doi:10.1016/j.ejc.2008.12.016.

108 Lev Telyatnikov, Guillermo Bernardez, Marco Montagna, Mustafa Hajij, Martin Carrasco, Pavlo Vasylenko, Mathilde Papillon, Ghada Zamzmi, Michael T. Schaub, Jonas Verhellen, Pavel Snopov, Bertran Miquel-Oliver, Manel Gil-Sorribes, Alexis Molina, Victor Guallar, Theodore Long, Julian Suk, Patryk Rygiel, Alexander Nikitin, Giordan Escalona, Michael Banf, Dominik Filipiak, Max Schattauer, Liliya Imasheva, Alvaro Martinez, Halley Fritze, Marissa Masden, Valentina Sánchez, Manuel Lecha, Andrea Cavallo, Claudio Battiloro, Matt Piekenbrock, Mauricio Tec, George Dasoulas, Nina Miolane, Simone Scardapane, and Theodore Papamarkou. TopoBench: A framework for benchmarking topological deep learning, 2025. arXiv:2406.06642.

109 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 6.1 edition, 2025. URL: https://doc.cgal.org/6.1/Manual/packages.html.

110 Julien Tierny, Guillaume Favelier, Joshua A. Levine, Charles Gueunet, and Michael Michaux. The topology toolkit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):832–842, 2017. doi:10.1109/TVCG.2017.2743938.

111 Raphaël Tinarrage. Computing persistent Stiefel–Whitney classes of line bundles. *Journal of Applied and Computational Topology*, 6(1):65–125, 2022. doi:10.1007/s41468-021-00080-4.

112 Lukáš Vokřínek. Decidability of the extension problem for maps into odd-dimensional spheres. *Discrete & Computational Geometry*, 57(1):1–11, 2017. doi:10.1007/s00454-016-9835-x.

113 Ichiro Yokota. On the cellular decompositions of unitary groups. *Journal of the Institute of Polytechnics, Osaka City University. Series A, Mathematics*, 7(1–2):39–49, 1956. https://projecteuclid.org/journals/journal-of-the-institute-of-polytechnics-osaka-city-university-series-a-mathematics/volume-7/issue-1-2/On-the-cellular-decompositions-of-unitary-groups/ojm/1353054782.full.

Mixup Barcodes: Quantifying Geometric-Topological Interactions Between Point Clouds

Hubert Wagner
University of Florida, Gainesville, FL, USA

Nickolas Arustamyan
University of Central Florida, Orlando, FL, USA

Matthew Wheeler
University of Florida, Gainesville, FL, USA

Peter Bubenik
University of Florida, Gainesville, FL, USA

Abstract

We propose a novel geometric-topological descriptor called a mixup barcode. Intuitively, it characterizes the shape of a point cloud as well as its spatial relationship with another point cloud embedded in the same ambient space. More technically, it enriches a standard persistence barcode with information on the image persistent homology. In three dimensions it captures natural spatial relationships like overlap and surrounding; in higher dimensions more intricate spatial relationships are captured. We provide a theoretical setup and a simple algorithm for mixup barcodes.

As a proof of concept, we explore data arising in a geometric-topological problem from machine learning. Specifically, we take first steps towards verifying a hypothesis stating that geometric-topological relationships within intermediate point cloud representations in an artificial neural network can hinder its training. More broadly, our experiments suggest that mixup barcodes are useful for characterizing spatial relationships and spatial interactions (i.e. the evolution of spatial relationships) that are hard to directly visualize or capture using standard methods.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Combinatorial algorithms

Keywords and phrases mixup barcode, persistent homology, persistence barcode, persistence diagram, image persistent homology, image persistence, deep learning, multilayer perceptron, topology of neural network embeddings, disentanglement

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.94

Related Version *Full Version*: `https://doi.org/10.48550/arXiv.2402.15058`

Supplementary Material *Software*: `https://github.com/hubwag/Mixup-SoCG26` [36]
archived at `swh:1:dir:9b75232380440dc7556b698dd1b7d07882a61fac`

Funding *Hubert Wagner*: Partially supported by the 2022 Google Research Scholar Award in Algorithms and Optimization.
Matthew Wheeler: Supported by The American Association of Immunologists through an Intersect Fellowship for Computational Scientists and Immunologists.

Acknowledgements The authors thank Ondřej Draganov, Teresa Heiss, Herbert Edelsbrunner and Yasu Hiraoka for helpful discussions.

© Hubert Wagner, Nickolas Arustamyan, Matthew Wheeler, and Peter Bubenik;
licensed under Creative Commons License CC-BY 4.0

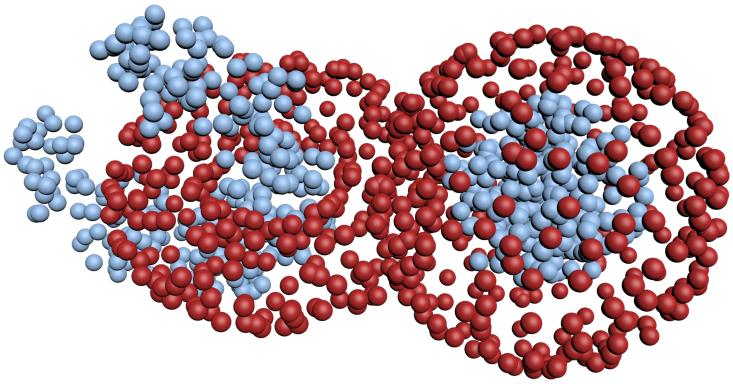


Figure 1 TDA allows us to characterize the shapes of the two, red and blue, point clouds. We also ask: what are the spatial relationships between these point clouds? Intuitively, a portion of the blue point cloud is surrounded by the spherical part of the red one; similarly, another portion of the blue point cloud loops around the toroidal subset of the red points. These are natural questions and we currently lack the tools (and in some cases the language) to discuss them. High-dimensional generalizations play an important role in the machine learning problem we consider.

1 Introduction

Computational geometry delivers a variety of tools for characterizing the shape of a point cloud. One can compute the convex hull, the Delaunay triangulation and its variants, and related filtrations such as alpha filtrations. Analogs for high dimensional point clouds such as Rips and Čech filtrations are becoming better understood [20] and more computable [4, 14].

The shape of data. In recent years, it has become popular to summarize the geometric-topological structure of a point cloud using persistent homology. It involves computing a persistence diagram, or barcode, serving as a geometric-topological descriptor of a dataset.

Beyond the shape of data. There are important problems which require characterizing the **spatial relationships** among datasets. We provide a new tool, called a **mixup barcode**, which captures both the shape of data and its spatial relationships with another dataset embedded in the same ambient space. See Figure 1 for an example. We will also consider spatial interactions, by which we mean the evolution of spatial relationships over time.

In overview, mixup barcodes capture how the inclusion of another point cloud affects the persistent homology of an existing point cloud. Specifically, we ask how the lifetime of each topological feature shortens in the presence of new data. This way we quantify the spatial relationships between the two point clouds. While this task fits the general framework of multi-parameter persistent homology, we take a more pragmatic and computable approach. We frame this intuition in terms of persistent homology as well its variant, image persistent homology, which was recently shown to be efficiently computable [7]. We carefully combine these two pieces of information.

A proof of concept application. Our method is motivated by a problem arising in machine learning. We will refer to this problem as *the disentanglement problem*. In [29], it was hypothesized that training can be hindered by geometric-topological interactions within the intermediate representations (point clouds) produced by such models. Intuitively, a

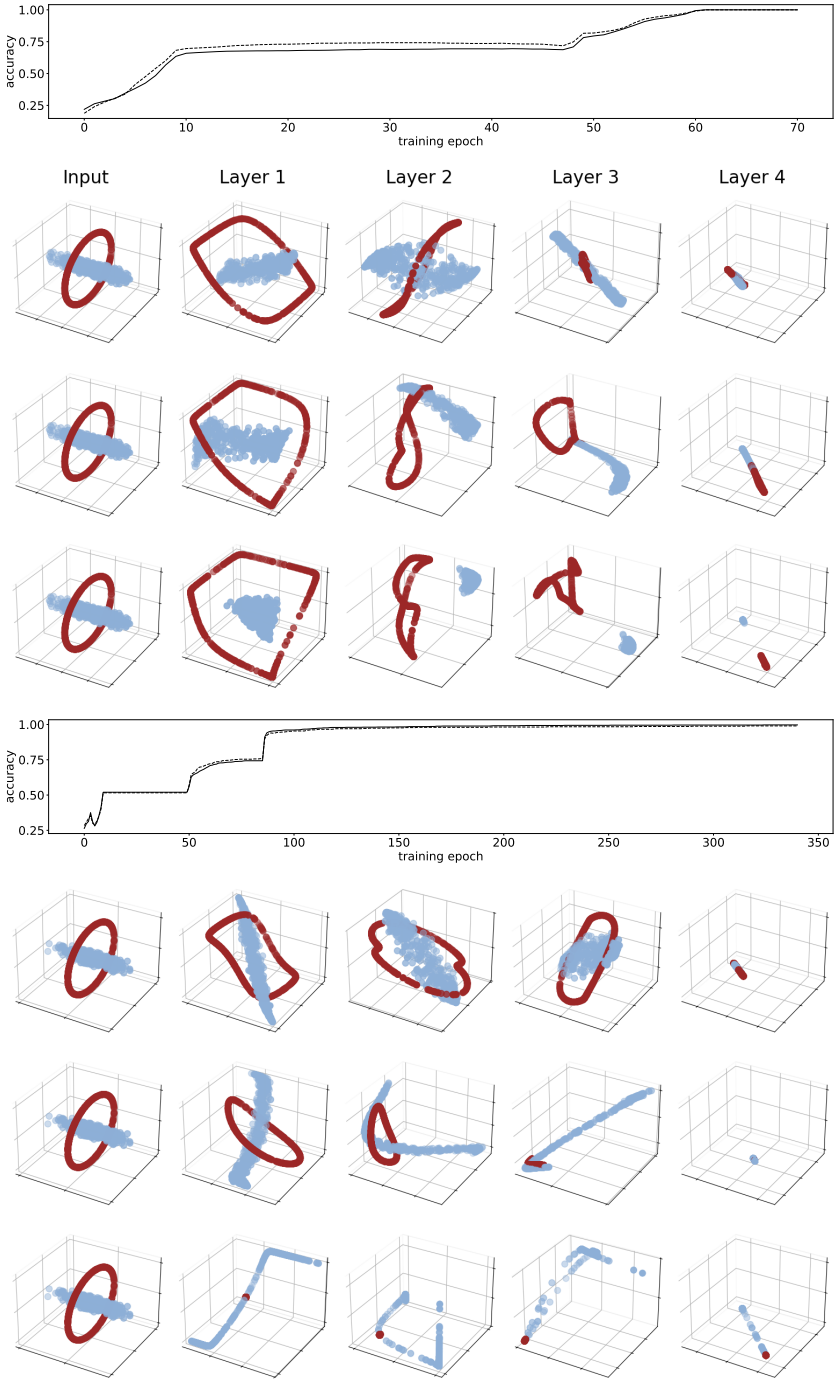


Figure 2 Two training runs of the same MLP classifier. For each run, three selected training epochs are shown as rows. Each row shows the outputs of consecutive layers from left to right. Intuitively, at the end of the training, the blue and red points output by the *last* layer should be separated. In both cases (almost) perfect test accuracy is achieved. However, in the second run the model failed to *disentangle* the red and blue points, and a small portion of the blue points became *confused* with the red points. In particular, this behavior lowers the generalization power of the model (since certain new inputs are bound to be misclassified). We propose tools for detecting such situations.

model may fail to **disentangle** the representations corresponding to data with different labels – see Figure 2 for a concrete example. First steps towards testing this hypothesis were recently made using persistent homology [27, 39]. We argue that our method offers additional, useful information, and present experiments supporting this claim. We elaborate on the experimental setup in Section 7.

Contributions. The contributions of this paper include:

1. Mathematical foundations for a new topological descriptor called a *mixup barcode* characterizing spatial relationships between point clouds.
2. Summary statistics quantifying the robustness of these relationships.
3. Experiments showing the computability and usefulness of mixup barcodes in the above-mentioned machine-learning problem.
4. A simple practical algorithm for computing mixup barcodes.
5. This is one of the first applications of image persistence [13] and the first application we are aware of for high-dimensional data.

Connections and potential impacts. We expect this work to expand the standard paradigm of topological data analysis, namely from summarizing and learning from the shape of a point cloud, to summarizing and learning from the spatial relationships among point clouds. This goal is aligned with the emerging direction of Chromatic Topological Data Analysis outlined in [16]. Our work was developed independently, but both directions build on the same theoretical and algorithmic foundation [13]. The main novelty of our approach comes from providing a single, compound descriptor that merges two simpler barcodes. These barcodes are typically considered in isolation – which loses important information that our method retains (at a negligible computational cost). We will contrast these approaches further in Section 8.

Once the theoretical and computational aspects are fully developed, the proposed tool will likely open new applications of geometric-topological methods.

2 Related work

We briefly review related work in computational geometry and topology, as well as their relevant applications in machine learning.

Image persistence. Regarding computational methods for image persistence, an algorithm was proposed by Cohen-Steiner and collaborators [13]. An implementation is available in the first version of Morozov's Dionysus library, but not in the current version. Recently, Bauer and Schmahl proposed a version [7] of the Ripser library for image persistence of Vietoris–Rips complexes, which informs our implementation. One of the first practical applications of image persistence was done recently in the context of image segmentation [33].

Cultrera di Montesano and collaborators [17] have proposed an approach for analyzing the mingling of a small number of 3-dimensional point clouds using image (as well as kernel and cokernel) persistence. One crucial development is the chromatic alpha complex which allows for the efficient computation of these variants of persistence for data embedded in low dimensions. Their motivation is similar to ours, although both projects were started and developed independently. Our focus is on a single descriptor combining standard persistence with image persistence, while theirs is on different types of persistent homology and connections between them. Natarajan et al., study the Morse theory of chromatic Delaunay complexes [28] and provide an implementation.

Induced matching. Our results are also related to work on induced matchings originating in [6]. In particular, similar ideas were independently explored by Gonzalez-Diaz and coauthors [22, 23, 35].

Persistent homology in machine learning. Techniques and tools from topological data analysis have recently found a number of fruitful applications to machine learning and in particular deep neural networks [44]. Persistent homology has been used in developing topological network layers [21, 24, 26]. Another approach incorporates topological information about the data into the loss function of a neural network [11, 12, 25, 37]. Topological methods were also used to analyze other aspects of artificial neural networks [9, 30, 32, 38, 40, 42]. In [3] a variant of persistent homology was developed to compare neural networks.

Disentanglement in deep neural networks. In machine learning, disentanglement is a broad concept. We are interested in how embeddings corresponding to different classes are separated during the training of a classifier. A motivation for this work goes back to an influential 2014 blog post [29] by Olah on "Neural Networks, Manifolds, and Topology" – ten years later we have appropriate computational tools to address this challenge.

Perhaps the fist explicit mention of (manifold) disentanglement in the context of deep learning is in the work of Brahma, Wu and Yiyuan [8]. The work of Zhou and collaborators provides a topological view on disentanglement in the context of machine learning as well as a thorough overview [43]. We refer the reader interested in the machine learning details there, and focus on geometric-topological aspects of the problem. Our experiments are inspired by the work by Naitzat and collaborators [27] and the follow-up by Wheeler, Bouza and Bubenik [39]. We alleviate some of the shortcomings of these approaches that stem from the limitations of persistent homology applied to a single point cloud. Reani and Bobrowski proposed a data-analysis method [31] based on matching geometrically close cycles.

3 Mixup barcodes

In this technical section we propose the mathematical setup for mixup barcodes. We delay detailed interpretation to the next section. We will work with filtrations of simplicial complexes, chain complexes, persistence modules and their decompositions.

Overview. In the simple case, we study point clouds A and B, via the inclusion $A \hookrightarrow A \cup B$. We ask how the persistence of each topological feature arising from A is shortened by the inclusion of points in B. The constructed *mixup barcode* splits each bar in the standard persistence barcode of A into two sub-bars. The first sub-bar quantifies this new, shortened lifetime, and is computed using a variant of persistent homology called image persistence. The second sub-bar is called a mixup sub-bar, and quantifies the shortening of the lifetime.

We follow up with technical definitions, but a reader interested in an intuitive explanation could jump to Section 5 and especially Figure 3.

Technical definitions. We will work with two filtrations L and K – which will be connected by an inclusion that we describe in a moment. Specifically, let K denote a filtered finite simplicial complex $K_1 \hookrightarrow K_2 \hookrightarrow \cdots \hookrightarrow K_n$ with inclusion maps $K_i \hookrightarrow K_j$ for $1 \leq i \leq j \leq n$, and similarly for L.

We assume that both filtrations are simplex-wise, i.e. for each i, L_{i+1} differs from L_i by at most one simplex and similarly for K. We further assume that each σ added in L_i, is not already present in K, namely $\sigma \notin K_{i-1}$, where $K_0 = \emptyset$. The index of a simplex is the i at

which it appears in the filtration, which is uniquely defined under the above assumptions. Further, we will sometimes assume that we have a real-valued order-preserving function f assigning a **filtration value** to each simplex.

The above assumptions may be met in the context of Vietoris–Rips filtrations, which will be our practical setup. In particular, while several simplices will typically appear at the same filtration value, it is easy to use an ordering of the simplices to obtain a finer simplex-wise filtration. We remark that everything in this section also generalizes to finite cell complexes.

The studied inclusion map. For each $k \geq 0$ we consider the inclusion map $\iota : L \hookrightarrow K$ between our filtrations L and K. This gives rise to a variety of algebraic structures and induced maps between them. Of primary interest are the persistence modules $H_k(L)$ and $H_k(K)$ and an induced map between them $H_k(\iota) : H_k(L) \to H_k(K)$. This information is summarized in the following commutative diagram of vector spaces and linear maps.

$$\begin{array}{ccccccccc} H_k(L_1) & \longrightarrow & H_k(L_2) & \longrightarrow & \dots & \longrightarrow & H_k(L_i) & \longrightarrow & \dots & \longrightarrow & H_k(L_n) \\ {\scriptstyle H_k(\iota_1)}\downarrow & & {\scriptstyle H_k(\iota_2)}\downarrow & & & & {\scriptstyle H_k(\iota_i)}\downarrow & & & & {\scriptstyle H_k(\iota_n)}\downarrow \\ H_k(K_1) & \longrightarrow & H_k(K_2) & \longrightarrow & \dots & \longrightarrow & H_k(K_i) & \longrightarrow & \dots & \longrightarrow & H_k(K_n) \end{array} \tag{1}$$

The image of the map of persistence modules in (1) is the following persistence module:

$$\mathrm{im}(H_k(\iota)) = \mathrm{im}(H_k(\iota_1)) \to \mathrm{im}(H_k(\iota_2)) \to \cdots \to \mathrm{im}(H_k(\iota_i)) \to \cdots \to \mathrm{im}(H_k(\iota_n))$$

We are interested in the barcodes of the persistence module $H_k(L)$ (i.e. the persistent homology of L) and the persistence module $\mathrm{im}(H_k(\iota))$ (i.e. the image persistent homology of the inclusion). Specifically, we will combine the information contained in these barcodes in a meaningful way, to form a single richer descriptor.

To do so, we notice that under our assumptions for each index $i \in \{1, 2 \dots, n\}$ there is at most one interval in the barcode with minimum value i. We call such i the **birth index** of the corresponding interval in the barcode. This allows us to match these barcodes by the birth *indices*; for brevity we refer to the following more general result.

▶ **Theorem 1** (Matching [6, Theorem 4.2, Proposition 5.7])**.** *There is a canonical surjective matching from the barcode of $H_k(L)$ to the barcode of $\mathrm{im}(H_k(\iota))$ which matches intervals with the same birth index.*

With this, we reorganize the information in this canonical matching of barcodes as follows.

▶ **Definition 2** (Mixup barcode)**.** *The **mixup barcode** in degree k of the inclusion $L \hookrightarrow K$ is the collection of triples of indices consisting of a triple (b, d', d) for each pair of matched intervals $[b, d)$ and $[b, d')$ in the barcodes of $H_k(L)$ and $\mathrm{im}(H_k(\iota))$; and a triple (b, b, d) for each unmatched interval $[b, d)$ in the barcode of $H_k(L)$.*

The above definition is in terms of the filtration indices. In practice, we often replace them with the filtration values of the corresponding simplex. Typically the distinction will be clear from context, but sometimes we will explicitly talk about **index mixup barcodes**.

Viewing the mixup barcode as a refinement of the persistence barcode of L, we visualize it in a similar way. Specifically, we plot the image sub-bar in a light color, and its complement, the mixup sub-bar, in a darker color. We consider a separate mixup barcode for each degree; see Figure 3 for an example.

Mixup barcodes for point clouds. We consider two finite point clouds, $A, B \subset \mathbb{R}^d$, or more generally, a finite metric space X with $A \subset X$ and $B = X \setminus A$, and construct Vietoris–Rips filtrations of A and $A \cup B$, with a common sequence of radii, $r_1 \leq r_2 \leq \cdots \leq r_n$. The simplicial filtrations L and K are given by $L_i = \mathrm{VR}(A; r_i)$ and $K_i = \mathrm{VR}(A \cup B; r_i)$. By the mixup barcode of $A \hookrightarrow A \cup B$, we mean the mixup barcode induced by $L \hookrightarrow K$ as defined above. Unlike a persistence barcode, a mixup barcode is sensitive to the *ordering* of the simplices, since it may affect the matching.

4 Related concepts and statistics

With the main technical definition in place, we explain the rationale behind it, and mention some properties. We start with defining an analog of a **persistence pair** [18] that can be expressed either in terms of the indices or filtration values.

▶ **Definition 3** (Mixup triple). *Each triple (b, d', d) in the mixup barcode of $L \hookrightarrow K$ is called an **mixup triple**.*

We reiterate what information is stored in each mixup triple.

▶ **Definition 4** (Persistence bar and image and mixup sub-bars). *For each mixup triple (b, d', d), we have a **persistence bar** $[b, d)$, which splits into what we call:*
- i *an **image sub-bar** $[b,\ d')$, and*
- ii *a **mixup sub-bar** $[d',\ d)$,*

noting that degenerate sub-bars of the form $[x, x)$ are allowed.

▶ **Observation 5.** *The mixup barcode of $L \hookrightarrow K$ contains the persistence bars of L and its image sub-bars correspond to the image persistence barcode. However, the mixup sub-bars will generally* not *coincide with any bars of kernel persistence barcode. We provide an explicit example in the full version of the paper.*

▶ **Definition 6** (Premature death). *Given an index mixup triple (b, d', d) we interpret the index d' as the index of premature death of the unique homology class γ born at L_b. Specifically, we mean the first index at which γ becomes trivial or merges with an older homology class in $H_k(L)$.*

▶ **Observation 7.** *We stress that a premature death arises from the image persistent homology vector spaces of our inclusion, namely $\frac{Z_k(L_i)}{B_k(K_j) \cap Z_k(L_i)}$. (Contrast this with the standard persistence: $\frac{Z_k(L_i)}{B_k(L_j) \cap Z_k(L_i)}$.) This premature death is therefore caused by the extra boundaries coming from K – but not by merging with another homology class present in $H_k(K)$ but absent from $H_k(L)$.*

4.1 Statistics

It is often useful to extract a single number from a persistence barcode. For example, the total persistence – the sum of the lengths of all bars – is used to quantify the topological complexity of a point cloud. In a similar vein, we define simple ways of extracting a number to quantify the strength of interactions between filtrations L and K.

▶ **Definition 8** (Mixup). *The mixup of a mixup triple (b, d', d) is the value $d - d'$.*

In other words, it is the length of the sub-bar (d', d) which quantifies the prematurity of the death of a single homology class.

▶ **Definition 9** (Total mixup). *The **total mixup** of $L \hookrightarrow K$ is the sum of the mixups for the bars in the mixup barcode.*

In our visualization, it is simply the sum of the lengths of the dark bars. We refer to this sum as total-mixup(A, B). Equivalently, it is the difference between the total persistence of A and the total image persistence of $A \hookrightarrow A \cup B$.

Scale invariant statistics. The next two definitions allow us to make scale-invariant measurements. Firstly, this allows us to compare data embedded in different dimensions. Secondly, the geometric scale of topological features is irrelevant for our application.

▶ **Definition 10** (Mixup percentage). *The **mixup percentage** of mixup triple $t = (b, d', d)$ is*

$$mixup_{\%}(t) = \frac{mixup(t)}{pers(t)} = \frac{d - d'}{d - b}. \tag{2}$$

▶ **Definition 11** (Total mixup percentage). *The **total mixup percentage** (**mean mixup percentage**) of a mixup barcode is the sum (mean) of the mixup percentages for all its bars.*

Note that the latter variant is between 0 and 1, but is *not* the ratio of the total mixup and the total persistence. The mean mixup percentage has the added benefit of allowing for the comparison between datasets of differing total persistence, which will be crucial in our application.

5 Geometric interpretation

In this section we provide intuitive interpretations of the information captured by mixup barcodes in various degrees. It turns out that they often capture natural spatial relationships.

Holes. We recall that for a shape embedded in three dimensional space, homology groups in **degree** 0, 1, 2 are often interpreted as various types of holes: the gaps between distinct parts of the object (or dually the connected components); tunnels (or handles); voids enclosed by the object. Note that in general voids correspond to H_{n-1} for objects in $\mathbb{R}^n$. Persistent homology captures the births and deaths of such features along the filtration.

Spatial relationships for point clouds. Going back to the case of a pair of point clouds A and B in $\mathbb{R}^3$, the shortening of the lifetime of a homological feature conveys how points from B fill the holes of different types.

Specifically, in degree 0, mixup quantifies to what extent the connected components in A *merge quicker* via points in B – which has natural interpretations as A and B **overlapping** or A being **separated** by B. In degree 1, it quantifies how much quicker points in B fill up a loop or tunnel formed by A – for example when a portion of A encircles a portion of B (like in a Hopf link). In degree 2, it quantifies how points in B contribute to filling up a void inside A – which we interpret as a portion of A **surrounding** a portion of B. Figure 3 illustrates these spatial relationships and the corresponding mixup barcode.

For the machine learning example illustrated in Figure 2, mixup in degree 1 is most relevant as it captures the red points initially *encircling* the blue points, and then either *tightening* around them, or *separating* from them. For the second case, mixup in degree 0 is also useful.

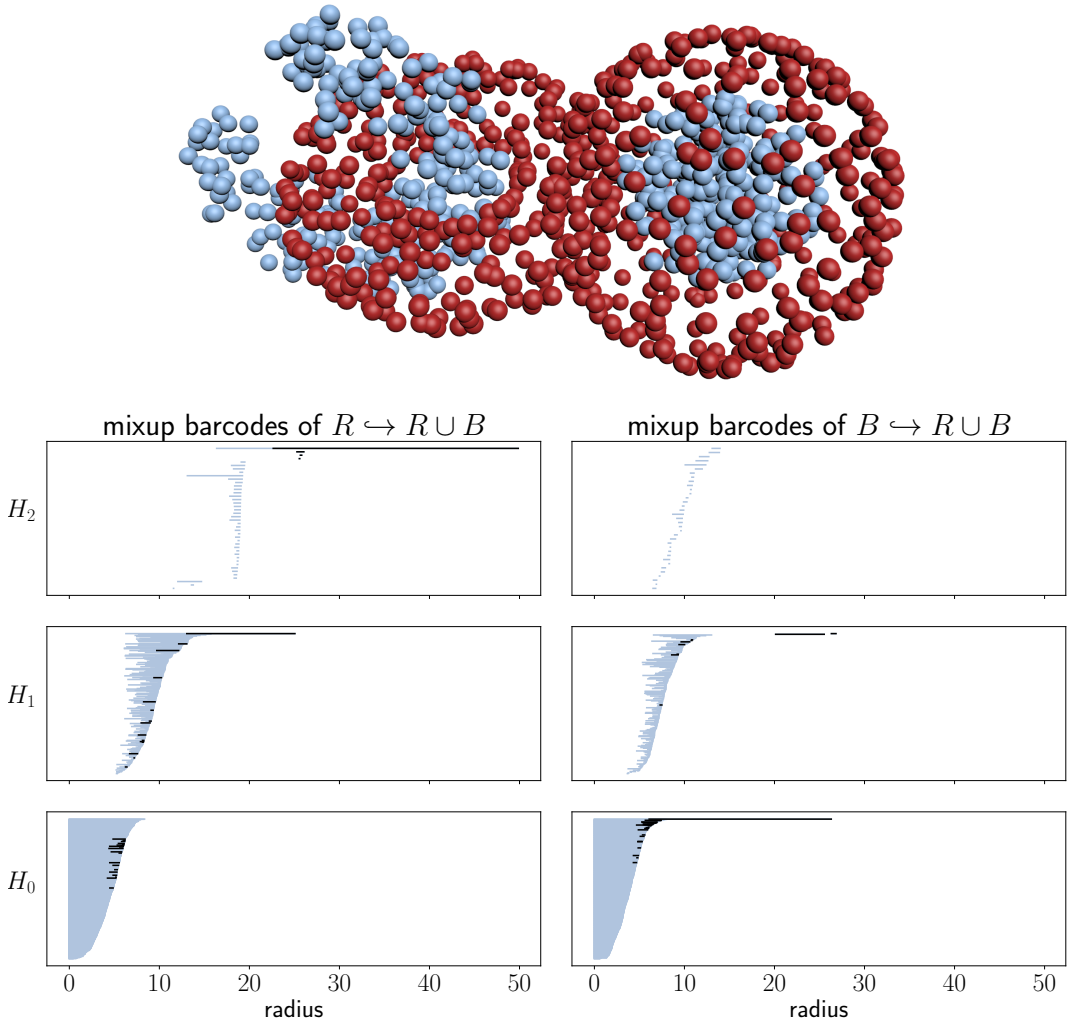


Figure 3 Top: Two point clouds: R red (dark) and B blue (light). **Bottom:** Mixup barcodes of R and B included in their union, in degrees $2, 1, 0$. Recall that each persistence bar is split into an image sub-bar and a mixup sub-bar (marked in thick black). **Bottom left:** In H_2, we see a prominent persistence bar with a long mixup bar. It detects the red **void** partially filled by blue points, interpreted as the red object **surrounding** a part of the blue object. Similarly in H_1 the prominent mixup bar detects the red handle (tunnel) that **encircles** the blue points. No significant mixup occurs in H_0, although the many shorter mixup bars detect the **overlap** between the two shapes. **Bottom right:** In H_1 the mixup barcode detects the handle **encircled** by the red object, with the late birth indicating that a large part is missing. The prominent bar for H_0 detects that the two blue connected components connect quicker via the red points. We interpret this as **separation** of the two blue parts by the red object. As expected, there is no prominent persistence in H_2.

In summary, for point clouds in $\mathbb{R}^n$, the mixup barcode captures: **overlap** or **separation** in degree 0 and **surrounding** in degree $n-1$. In remaining degrees it captures how holes in degree k present in L are filled by boundaries in K, which corresponds to more subtle and harder to name interactions.

6 Algorithm

In this section we describe a simple algorithm for the decomposition of a persistence barcode into a mixup barcode and prove its correctness under the assumptions in Section 3. We start by recalling the algorithms for standard and image persistent homology, which will be part of our construction.

Standard reduction algorithm. Persistent homology can be computed using a version of Gaussian elimination on the boundary matrix representing the filtration [18]. We call such an algorithm REDUCE.

The ordering of columns and rows corresponds to the order in which the elements (typically simplices) appear in the filtration. By adding the columns (only from left to right) modulo 2, the algorithm removes conflicts between columns, by which we mean columns having the same pivot. The index persistence pairs are formed by the indices of nonzero columns and their pivots in the final (reduced) matrix. This information does not depend on the choice of column additions [19].

Image persistent homology computations. Image persistent homology can be computed using a variant of the above algorithm, as described in [13, pp. 5-6]. Perhaps surprisingly, this algorithm reorders the *rows* of the boundary matrix, so that the elements from L come first (ordered by the filtration index), followed by the elements from $K \setminus L$ (also ordered by the filtration index). The image index persistence pairs are of the form (σ, τ'), where the τ' is the index of a nonempty column whose pivot σ corresponds to a simplex in L [13, Observations (ii),(iii)]. This yields a modified Elder rule: The youngest class from $K \setminus L$ is destroyed whenever available; otherwise, the youngest class from L is destroyed, as usual [18]. A cohomological version is obtained by working on anti-transposed matrices [5].

Mixup barcodes algorithm. Algorithm 1 outlines the algorithm that decomposes the bars in the standard persistence diagram into the mixup barcode. The key new idea is to coordinate the two matrix reductions, but preparing two filtrations with consistent indexing. These reductions yield the standard and image persistent homology. The latter computation involves the reordering of the rows, as mentioned above. This coordinated reduction allows us to merge the two pieces of information. In contrast, computing the standard and image persistence on two unrelated filtrations loses the correspondence between these two pieces of information. Additionally, we reiterate that mixup bars generally do not coincide with kernel persistence, necessitating the use of our algorithm.

Computational complexity and implementation. Due to the use of Gaussian elimination, the algorithm runs in $O(n^3)$ worst-case time, where n is the number of simplices. As with standard persistence [5], the running time is significantly better in practice.

Algorithm 1 Algorithm for Mixup Barcode in degree k.

Require: Simplex-wise filtrations L and K as in Section 3; f assigning the filtration value to each simplex; degree k

BK = matrix with columns containing the boundaries of the k- and $(k+1)$-simplices of K with rows and columns ordered by the filtration index
reorder the rows of BK such that the simplices from L have smaller indices than the simplices of $K \setminus L$ while retaining the relative order of simplices in L and also in $K \setminus L$
form matrix BL by zeroing out the entries in BK involving the simplices of $K \setminus L$
REDUCE matrices BL and BK.
for each index σ of a k-simplex in L such that $BL[\sigma] = 0$ **do**
    $birth = f(\sigma)$; $death = death' = \infty$
    if $\exists\ \tau$ such that $\text{pivot}(BL[\tau]) = \sigma$ **then** $death = f(\tau)$
    if $\exists\ \tau'$ such that $\text{pivot}(BK[\tau']) = \sigma$ **then** $death' = f(\tau')$
    record mixup triple $(birth, death', death)$
return all the recorded mixup triples

▶ **Theorem 12** (Mixup Decomposition Theorem)**.** *The algorithm in Algorithm 1 returns a correct mixup barcode of $L \hookrightarrow K$.*

Proof is in the Appendix of the full version of the paper.

7 Experiments

We apply our method to analyzing high-dimensional point clouds. They come from a geometric problem arising in machine learning. The goal is to gain insights about the training of an artificial neural network [2], the **multilayer perceptron** (MLP). It is the simplest *deep learning* method, often used for classification. We remark that models based solely on MLPs can achieve state of the art performance on challenging tasks [34].

Model parameters. We use a 5-layer MLP, with embedding dimensions $512, 256, 128, 10$ and L, where L depends on the number of considered labels (in our case 3 or 10). The model has $565,248 + 10L$ trainable parameters (weights).

7.1 Datasets

We use two well known datasets with different levels of complexity. Note that while performing classification on these datasets is considered solved – understanding how machine learning models achieve their good performance is an open problem.

MNIST. This is a standard dataset, containing 60,000 grayscale images of size 28×28 depicting handwritten decimal digits. It is a heavily preprocessed version of real-world images, so that the Euclidean distance between two images closely corresponds to their perceptual dissimilarity. For this reason, representing the images as a (labeled) point cloud in $\mathbb{R}^{784}$ yields a faithful representation of the data. It comes split into 50,000 training images, and 10,000 test images. We normalize the values to be between 0 and 1. The model achieves 99% test accuracy when trained and tested on the images with labels 0,1 and 2, which we then use for our experiments. These are images of handwritten zeros, ones and twos. It is known to be an easy problem so we expect to see low mixup at the end of training. We are however curious about the higher-dimensional embeddings coming from the intermediate layers.

CIFAR10. Another standard dataset, containing 50,000 color photos of size 32×32 depicting various objects, animals etc. We treat this data as a (labeled) point cloud in $\mathbb{R}^{3072}$, but the Euclidean distance fails to capture the perceptual similarity. We expect to see more significant entanglement between different classes. This dataset is quite challenging for a basic MLP which achieves only 76% test accuracy when trained and tested the images with labels 0, 1 and 2. These are images of airplanes, cars and birds. We expect it to fail to disentangle the data, especially differentiating between airplanes and birds may be hard due to similar backgrounds.

7.2 Experiments

We use the new method to analyze geometric data arising from a machine learning model. The goal is to quantify geometric-topological difficulties during training – as theoretically outlined by Olah [29]. The results highlight the usefulness of the new methodology – especially in comparison to existing applications of persistent homology [27,39] that focused on quantifying the shape of a single point cloud.

In geometric terms, such a model learns a sequence of linear transformations each followed by a projection onto the positive orthant of $\mathbb{R}^m$, where m may be different at each step. This is done in an iterative training process, whose goal is to finally embed each input point with label number q near the q-th standard basis vector.

Goals. We consider the intermediate representation of the data produced by the i initial layers of the model, namely $M_k = \ell_k \circ \ell_{k-1} \circ ... \circ \ell_1$. Given a single input vector v, we consider $(M_k(v))_k$, namely the sequence of embeddings of v produced by each M_k. Extending this to the entire input point cloud, we get a sequence of point clouds – which additionally change as training progresses. Figure 2 illustrates this for a simplified three-dimensional case.

> We analyze how the representation of input data disentangles passing through the layers – and track this process during training. We aim to verify the conjecture that complicated spatial interactions can hinder training.

We return to the training depicted in Figure 2. In one of the two cases, the model failed to fully disentangle the two point clouds, even though both training and test accuracy reach almost 100%. Such a model is problematic. In this particular case, consider new examples corresponding to the blue region around which the red loop was constricted. These examples are likely to be misclassified, and can for example be exploited in an adversarial attack. More generally, this example shows that geometric-topological information about the embeddings can reveal information about the quality of the training, even when the accuracy (and by analogy the value of the loss function) suggest no such problems.

This situation is of course idealized: the intermediate embedding dimensions can be arbitrarily high, which simplifies the disentanglement problem. Still, complicated entanglement is hypothesized [29, 43] to correlate with training difficulties such as over-fitting and lack of generalization. Indeed, even if disentanglement is theoretically possible, it does not mean that the simplistic training process (oblivious to the global geometric-topological relationships) can realize it. Case in point: in our example, the problem was solvable in this dimension, but one of the training runs failed.

Datasets. To train the model, we will be using two datasets MNIST (easier, 99% accuracy) and CIFAR10 (harder, 76% accuracy). See Section 7.1 for more details. After training, we consider the embeddings produced from the test data. We treat the datasets fa finite labeled point cloud in $X \subset \mathbb{R}^d$. By X_q we mean **class** q of dataset X, namely the subset of points with label number q.

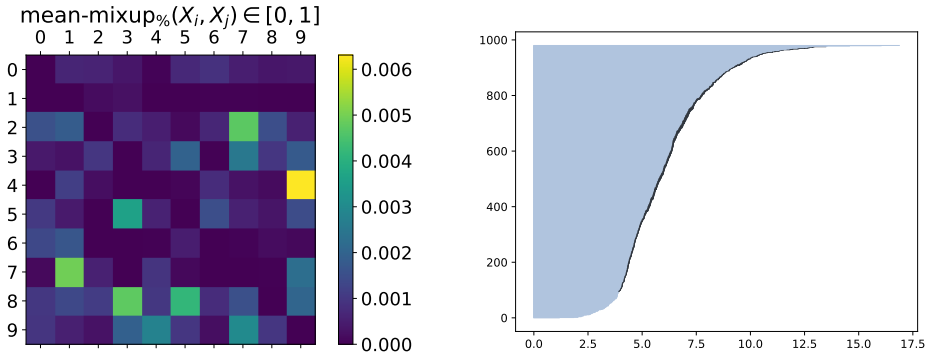


Figure 4 Left: The mean mixup percentage (in degree 0) between all pairs of MNIST classes. Most values are low, as expected due to the simplicity of this data. **Right**: The mixup barcode between the images of fours and nines, which have the greatest mixup. Many bars with small – but positive – mixup suggest that the two point clouds overlap along a broad but shallow interface.

Computations and subsampling. Due to space restrictions we elaborate on these aspects in the full version of the paper.

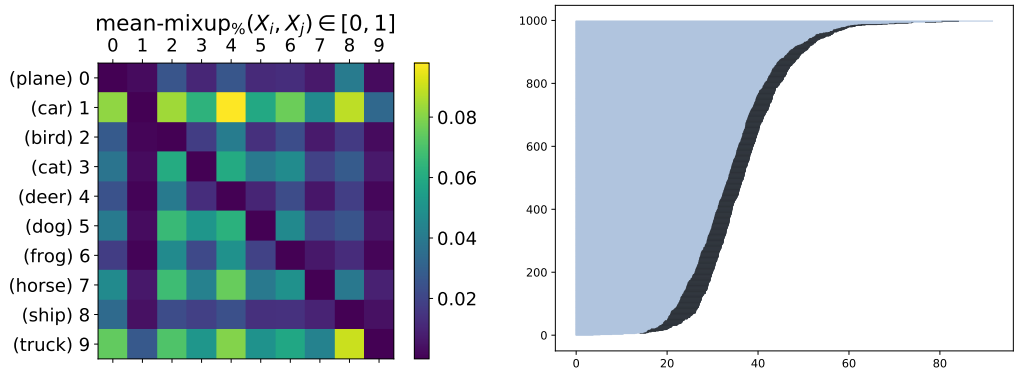


Figure 5 Left: The mean mixup percentage (in degree 0) between all pairs of CIFAR10 classes. Note that the values are generally an order of magnitude higher than for the other dataset, with the highest value of 0.09 between classes 1 (car) and 4 (deer). **Right:** The mixup barcode between classes 1 and 4, suggesting a significantly more robust overlap than we saw in the other dataset.

7.3 Warmup: mixup barcodes on raw inputs

While our primary focus is to analyze the model (and not the input data), we focus on the raw MNIST data for a moment. We do this because previous studies considered the disentanglement of different classes of this dataset [39] through the lens of persistent homology.

However, strong performance of (non-kernel) linear classifiers on MNIST [15, 41] suggests that pairs of point clouds belonging to different classes are easy to separate using an affine hyperplane. This suggests that they are unlikely to be strongly entangled.

As a sanity check, we verified that despite complicated topology, the mixup (which we use as a proxy for entanglement) is typically low. Specifically, we compute the mean percentage mixup in degree 0 for all pairs of classes of examples, namely for $A = X_i$ and $B = X_j$ for all $0 \leq i, j < 10$. The results for MNIST and CIFAR10 are in Figures 4 and 5.

As expected, CIFAR10 exhibits greater mixup, often by an order of magnitude. It is easy to see that persistence barcodes do not distinguish between the two cases. (The only difference comes from different dimensions of the two datasets.)

We emphasize that in general persistent homology of the data may be complicated, even if subsets of point clouds do not exhibit complex spatial relationships. Using persistence directly may overestimate the amount of entanglement. The extra information present in the mixup barcodes is therefore crucial for robust characterization of such phenomena.

7.4 Characterizing the disentanglement across layers

Our goal is to track disentanglement across layers during training. Specifically, we will analyze a labeled point cloud $X_{kqt} \subset \mathbb{R}^n$ for each layer k ($1 \leq k \leq 5$), training epoch t, and label q ($1 \leq q \leq 3$). The information about disentanglement will be characterized using the following concept.

▶ **Definition 13** (Mixup profile). *At each layer k and training epoch t, the mixup profile is computed as: $P_{kt} = \max_j\{\textit{total-mixup}_{\%}(X_{kjt}, \bigcup_{r\neq j} X_{krt})\}$, in homological degrees 0 and 1.*

The **mixup profile**, P_{kt}, measures complexity and strength of interactions between data representations with different labels, serving as a proxy for entanglement. It is visualized as a surface plot, with k (layer) and t (time) on the horizontal axes and P_{kt} (strength of spatial relationship) as the height.

Outcomes. See Figures 6 and 7 for a direct visualization of the mixup profiles for the MNIST and CIFAR datasets.

The main observation is that the mixup profiles are significantly different for the intermediate representations of the two datasets, suggesting that the interactions captured by the mixup barcodes indeed correlate with training difficulties.

Outcomes summary. The mixup profiles successfully distinguish between the training on the CIFAR and MNIST datasets. In particular, the mixup quickly drops for the easier dataset (MNIST), and not for the harder dataset (CIFAR). Overall, the statistics derived from the mixup barcodes do correlate with difficulties during training. They therefore support the conjecture stated in [29]. We stress that while MNIST and CIFAR10 are simple datasets, we analyze the training process on these datasets – which is poorly understood, even for these datasets. We discuss this further momentarily.

8 Discussion

We presented a novel geometric-topological descriptor called a mixup barcode, and its summary statistics. Unlike persistence barcodes, mixup barcodes can be used to characterize not only the shape of a dataset but also its spatial relationships with other datasets. While developed independently, it complements recent development of **Chromatic TDA**, an emerging branch of computational geometry and topology [16]. One advantage of our approach is that a single barcode is produced, which is often simpler to explain and analyze, especially compared to the six barcodes in [16]. Additionally, the mixup barcode retains the

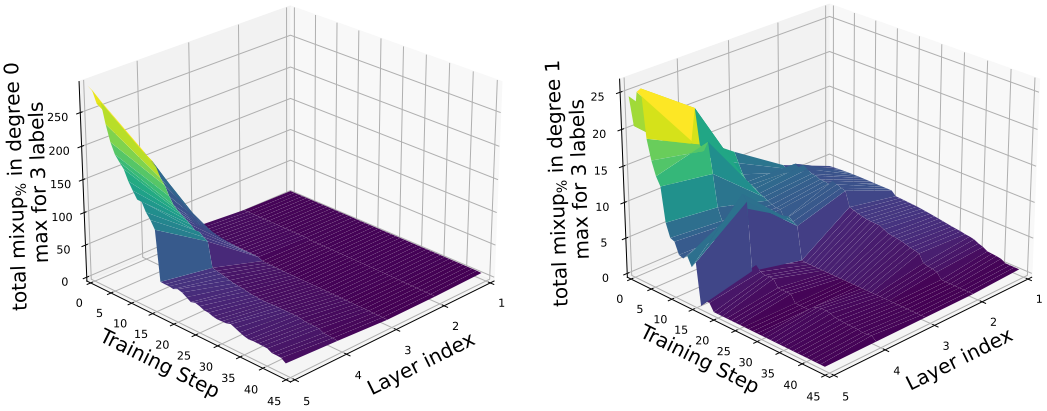


Figure 6 Mixup profiles characterizing the training on the MNIST dataset, in degree 0 (**left**) and degree 1 (**right**). We recall that the height corresponds to the mixup – the complexity of geometric-topological relationships – between intermediate embeddings of data with different labels at a given training step and layer of the model. The point closest to the viewer corresponds to the last layer at the end of training. Here, the mixup initially *grows* with consecutive layers (which makes sense since the input data is simple, but the matrix in each layer is initially random). On the other hand, mixup drops throughout with successive training iterations. This reflects that the network successfully *disentangles* all the embeddings.

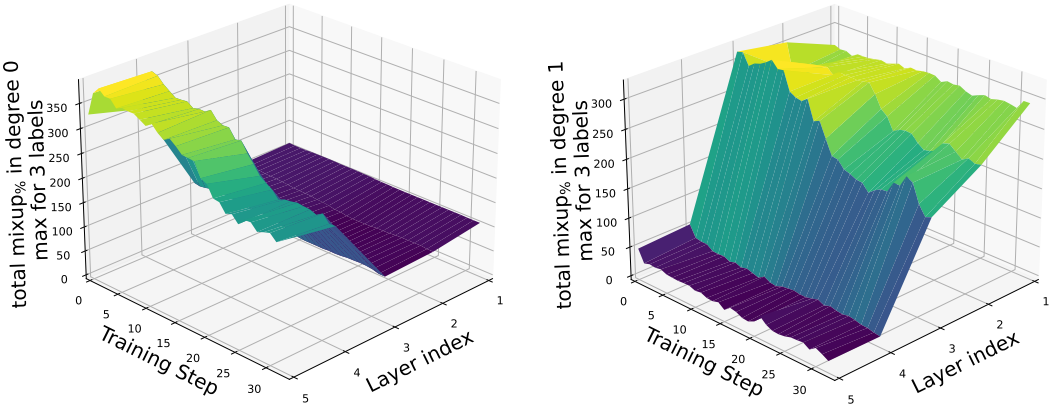


Figure 7 Mixup profiles characterizing the training on the more challenging CIFAR dataset. While the total mixup percentage generally drops during training, it remains generally much higher for CIFAR compared to MNIST. This is consistent with the worse performance of this model on the CIFAR dataset. In particular, for the last layer, mixup in degree 0 initially drops but stabilizes at a relatively high value (which is consistent with the partially successful training); in degree 1 it remains constant (and much higher than for MNIST) – this hints at more complicated spatial relationships between inputs with different classes that were hard to disentangle. Overall, the graphs suggest that the information captured in the mixup barcodes does indeed correlate with training difficulties and the model's failure to disentangle the embeddings.

crucial matching (between the two relevant barcodes), which is otherwise lost. Note that this information cannot in general be reconstructed using individual barcodes, expressed in terms of filtration values. Indeed, the proposed algorithm, working at the level of *index* persistence of two *consistently indexed* filtrations, is necessary.

We also reiterate that kernel persistence bars capture different information than our mixup sub-bars. For our application, it was crucial to use the total mixup percentage, which relies on the matching between persistence and image bars.

Our experiments mark the first application of Chromatic TDA ideas to analyzing high-dimensional data – as well as the first practical application of image persistence to such data. We believe image persistence (and related variants) are powerful tools, but currently underdeveloped and underutilized. We expect that other pairs (or perhaps larger subsets) of barcodes can be combined in a similar way.

On the applied side, this new tool sheds light on the complicated geometry and topology of the training process of machine learning models. The experiments suggest that the theoretical considerations proposed in [29] (in terms of smooth manifolds, point-set topology and knot theory) can now be verified computationally (on finite point clouds). The main outcome is that mixup barcodes capture spatial interactions that were conjectured to hinder training [29].

We expect that this new tool could help detect problems – such as overfitting – already during training, and regularize (or otherwise better control) the training to avoid such problems altogether.

However, further progress, including more detailed experiments, is limited by underdeveloped computational tools and theory. In particular a vectorization, for example generalizing persistence landscapes [10] or images [1], would allow us to feed the mixup barcodes into a statistical or machine learning pipeline. Relatively slow computation [7] of image persistence is another bottleneck. For now we are limited to simple descriptive statistics on small data.

Still, we do believe that our preliminary experiments hint at the usefulness of mixup barcodes, in particular in the context of high-dimensional data.

References

1 Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017. URL: `https://jmlr.org/papers/v18/16-337.html`.

2 Shunichi Amari. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, EC-16(3):299–307, 1967. `doi:10.1109/PGEC.1967.264666`.

3 Serguei Barannikov, Ilya Trofimov, Nikita Balabin, and Evgeny Burnaev. Representation topology divergence: A method for comparing neural network representations. *arXiv preprint arXiv:2201.00058*, 2021.

4 Ulrich Bauer. Ripser: efficient computation of vietoris–rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423, 2021. `doi:10.1007/S41468-021-00071-5`.

5 Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat: Persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78:76–90, 2017. Algorithms and Software for Computational Topology. `doi:10.1016/j.jsc.2016.03.008`.

6 Ulrich Bauer and Michael Lesnick. Induced matchings and the algebraic stability of persistence barcodes. *Journal of Computational Geometry*, 6(2):162–191, 2015. `doi:10.20382/JOCG.V6I2A9`.

7 Ulrich Bauer and Maximilian Schmahl. Efficient computation of image persistence. In *39th International Symposium on Computational Geometry (SoCG 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPIcs.SoCG.2023.14`.

8 Pratik Prabhanjan Brahma, Dapeng Wu, and Yiyuan She. Why deep learning works: A manifold disentanglement perspective. *IEEE Transactions on Neural Networks and Learning Systems*, 27(10):1997–2008, 2016. `doi:10.1109/TNNLS.2015.2496947`.

9 Rickard Bruel Gabrielsson and Gunnar Carlsson. Exposition and interpretation of the topology of neural networks. *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*, pages 1069–1076, 2019. ISBN: 9781728145495. arXiv:1810.03234, doi:10.1109/ICMLA.2019.00180.

10 Peter Bubenik et al. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.*, 16(1):77–102, 2015. doi:10.5555/2789272.2789275.

11 Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. A topological regularizer for classifiers via persistent homology. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2573–2582. PMLR, 16–18 April 2019. URL: https://proceedings.mlr.press/v89/chen19g.html.

12 James R. Clough, Nicholas Byrne, Ilkay Oksuz, Veronika A. Zimmer, Julia A. Schnabel, and Andrew P. King. A Topological Loss Function for Deep-Learning Based Image Segmentation Using Persistent Homology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):8766–8778, December 2022. doi:10.1109/TPAMI.2020.3013679.

13 David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Dmitriy Morozov. Persistent homology for kernels, images, and cokernels. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 1011–1020. SIAM, 2009. doi:10.1137/1.9781611973068.110.

14 Matija Čufar. Ripserer. jl: flexible and efficient persistent homology computation in julia. *Journal of Open Source Software*, 5(54):2614, 2020. doi:10.21105/JOSS.02614.

15 Dennis DeCoste and Dominic Mazzoni. Fast query-optimized kernel machine classification via incremental approximate nearest support vectors. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 115–122, Washington, DC, USA, 2003. AAAI Press. Includes experiments on all 45 pairwise MNIST digit classification tasks; Table 1 reports linear SVM error rates on raw 784-D pixels for several pairs (e.g., 0 vs 1, 8 vs 3, 2 vs 5). URL: http://www.aaai.org/Library/ICML/2003/icml03-018.php.

16 Sebastiano Cultrera di Montesano, Ondrej Draganov, Herbert Edelsbrunner, and Morteza Saghafian. Chromatic topological data analysis. *arXiv preprint arXiv:2406.04102*, 2024. doi:10.48550/arXiv.2406.04102.

17 Sebastiano Cultrera di Montesano, Ondřej Draganov, Herbert Edelsbrunner, and Morteza Saghafian. Chromatic alpha complexes. *Foundations of Data Science*, 8:30–62, 2026.

18 Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

19 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, pages 454–463. IEEE, 2000. doi:10.1109/SFCS.2000.892133.

20 Herbert Edelsbrunner and János Pach. Maximum betti numbers of čech complexes. In *40th International Symposium on Computational Geometry (SoCG 2024)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SoCG.2024.53.

21 Rickard Brüel Gabrielsson, Bradley J. Nelson, Anjan Dwaraknath, and Primoz Skraba. A Topology Layer for Machine Learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1553–1563. PMLR, 26–28 August 2020. URL: https://proceedings.mlr.press/v108/gabrielsson20a.html.

22 R. Gonzalez-Diaz, M. Soriano-Trigueros, and A. Torras-Casas. Partial matchings induced by morphisms between persistence modules. *Computational Geometry*, 112:101985, 2023. doi:10.1016/j.comgeo.2023.101985.

23 Rocío González-Díaz, Manuel Soriano-Trigueros, and Álvaro Torras-Casas. Additive partial matchings induced by persistence morphisms. *arXiv preprint*, 2020. arXiv:2006.11100.

24 Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 1633–1643, Red Hook, NY, USA, 2017. Curran Associates Inc.

25 Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. *Advances in neural information processing systems*, 32, 2019.

26 Kwangho Kim, Jisu Kim, Manzil Zaheer, Joon Sik Kim, Frederic Chazal, and Larry Wasserman. Pllay: Efficient topological layer based on persistence landscapes. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

27 Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. Topology of deep neural networks. *The Journal of Machine Learning Research*, 21(1):7503–7542, 2020.

28 Abhinav Natarajan, Thomas Chaplin, Adam Brown, and Maria-Jose Jimenez. Morse theory for chromatic delaunay triangulations. *arXiv preprint*, 2024. arXiv:2405.19303.

29 Chris Olah. Neural Networks, Manifolds, and Topology, 2014. Accessed: 2024-06-28. URL: http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/.

30 Archit Rathore, Nithin Chalapathi, Sourabh Palande, and Bei Wang. TopoAct: Exploring the Shape of Activations in Deep Learning. *arXiv: 1912.06332*, pages 1–16, 2019. arXiv:1912.06332.

31 Yohai Reani and Omer Bobrowski. Cycle registration in persistent homology with applications in topological bootstrap. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5579–5593, 2022. doi:10.1109/TPAMI.2022.3217443.

32 Bastian Rieck, Matteo Togninalli, Christian Bock, Michael Moor, Max Horn, Thomas Gumbsch, and Karsten Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. In *International Conference on Learning Representations*, 2018.

33 Nico Stucki, Johannes C Paetzold, Suprosanna Shit, Bjoern Menze, and Ulrich Bauer. Topologically faithful image segmentation via induced matching of persistence barcodes. In *International Conference on Machine Learning*, pages 32698–32727. PMLR, 2023. URL: https://proceedings.mlr.press/v202/stucki23a.html.

34 Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. MLP-mixer: An all-MLP architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

35 Álvaro Torras-Casas, Eduardo Paluzo-Hidalgo, and Rocío González-Díaz. Topological data quality via 0-dimensional persistence matchings. *arXiv preprint*, 2023. arXiv:2306.02411.

36 Hubert Wagner, Nickolas Arustamyan, Matthew Wheeler, and Peter Bubenik. hubwag/Mixup-SoCG26. Software, swhId: swh:1:dir:9b75232380440dc7556b698dd1b7d07882a61fac (visited on 2026-05-18). URL: https://github.com/hubwag/Mixup-SoCG26, doi:10.4230/artifacts.26095.

37 Fan Wang, Huidong Liu, Dimitris Samaras, and Chao Chen. TopoGAN : A Topology-Aware Generative Adversarial Network. In *European Conference on Computer Vision*, pages 118–136. Springer, 2020. doi:10.1007/978-3-030-58580-8_8.

38 Satoru Watanabe and Hayato Yamana. Topological measurement of deep neural networks using persistent homology. *Annals of Mathematics and Artificial Intelligence*, 90(1):75–92, January 2022. doi:10.1007/s10472-021-09761-3.

39 Matthew Wheeler, Jose Bouza, and Peter Bubenik. Activation landscapes as a topological summary of neural network performance. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 3865–3870. IEEE, 2021. doi:10.1109/BIGDATA52589.2021.9671368.

40 Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, and Chao Chen. Link Prediction with Persistent Homology: An Interactive View. In *International conference on machine learning*, pages 11659–11669. PMLR, 2021. URL: http://proceedings.mlr.press/v139/yan21b.html.

41 Kai Yu, Tong Zhang, and Yihong Gong. Improved local coordinate coding using local tangents. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 1215–1222, 2010. Table 1 reports a 12.0% error rate for a linear SVM trained directly on raw MNIST images.

42 Songzhu Zheng, Yikai Zhang, Hubert Wagner, Mayank Goswami, and Chao Chen. Topological detection of trojaned neural networks. *Advances in Neural Information Processing Systems*, 34:17258–17272, 2021. URL: `https://proceedings.neurips.cc/paper/2021/hash/8fd7f981e10b41330b618129afcaab2d-Abstract.html`.

43 Sharon Zhou, Eric Zelikman, Fred Lu, Andrew Y Ng, Gunnar E Carlsson, and Stefano Ermon. Evaluating the disentanglement of deep generative models through manifold topology. In *International Conference on Learning Representations*, 2020.

44 Ali Zia, Abdelwahed Khamis, James Nichols, Zeeshan Hayder, Vivien Rolland, and Lars Petersson. Topological Deep Learning: A Review of an Emerging Paradigm, February 2023. arXiv:2302.03836 [cs]. `doi:10.48550/arXiv.2302.03836`.

An Optimal Algorithm for Computing Many Faces in Line Arrangements

Haitao Wang
Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA

Abstract

Given a set of m points and a set of n lines in the plane, we consider the classical problem of computing the faces of the arrangement of the lines that contain at least one point. We present an algorithm of $O(m^{2/3}n^{2/3} + (n+m)\log n)$ time for the problem. We also prove that this matches the lower bound under the algebraic decision tree model and thus our algorithm is optimal. In particular, when $m = n$, the runtime is $O(n^{4/3})$, which matches the worst case combinatorial complexity $\Omega(n^{4/3})$ of all output faces. This is the first optimal algorithm since the problem was first studied more than three decades ago [Edelsbrunner, Guibas, and Sharir, SoCG 1988].

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases Many faces, line arrangements, cuttings, Γ-algorithms, decision tree complexities

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.95

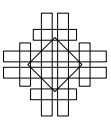
Related Version *Full Version*: https://arxiv.org/abs/2603.04863

1 Introduction

Let P be a set of m points and L a set of n lines in the plane. We consider the problem of computing the faces of the arrangement of the lines of L that contain at least one point of P; these faces are called *non-empty faces*. Note that each non-empty face only needs to be output once if it contains multiple points of P. Let $\mathcal{A}(L)$ denote the arrangement of L.

Previous work. This is one of the most fundamental problems in computational geometry and has been studied extensively. A straightforward algorithm can implicitly determine the non-empty faces of $\mathcal{A}(L)$ in $O(m\log n + n^2)$ time, by first constructing $\mathcal{A}(L)$, building a point location data structure for $\mathcal{A}(L)$ [15, 21, 25], and then finding the non-empty faces using the point location queries for points of P. The following upper bounds have been proved on the combinatorial complexity of all non-empty faces of $\mathcal{A}(L)$: $O(m^{2/3}n^{2/3}+n)$ [10], $O(n\sqrt{m})$ [17], and $O(n+m\sqrt{n})$ [17]. A lower bound of $\Omega(m^{2/3}n^{2/3}+n)$ [17] is also known, which matches the $O(m^{2/3}n^{2/3}+n)$ upper bound in [10]. Note that the minimum of the above upper bounds is at most $O(m\log n + n^2)$ regardless of the values of m and n. Hence, using the above straightforward approach, $O(m\log n + n^2)$ time is also sufficient to output all non-empty faces explicitly. To have a more efficient algorithm, Edelsbrunner, Guibas, and Sharir [14] first studied the problem and gave a randomized algorithm of $O(m^{2/3-\delta}n^{2/3+2\delta}\log n + n\log n\log m)$ expected time, for any $\delta > 0$. Subsequently, an improved deterministic algorithm of $O(m^{2/3}n^{2/3}\log^{5/3} n\log^{1.11}(m/\sqrt{n}) + (m+n)\log n)$ time was proposed by Agarwal [1]. Also, Agarwal, Matoušek, and Schwarzkopf [2] presented a randomized algorithm of $O(m^{2/3}n^{2/3}\log(n/\sqrt{m}) + (m+n)\log n)$ expected time. No progress had been made for a while until Wang [26] revisited the problem in SODA 2022 and derived a new deterministic algorithm of $O(m^{2/3}n^{2/3}\log^{2/3}(n/\sqrt{m}) + (m+n)\log n)$ time.


© Haitao Wang;
licensed under Creative Commons License CC-BY 4.0



On the other hand, $\Omega(m^{2/3}n^{2/3} + n\log n + m)$ is a lower bound for solving the problem due to the above $\Omega(m^{2/3}n^{2/3} + n)$ lower bound [17] on the combinatorial complexity of all non-empty faces and also because computing a single face in line arrangements requires $\Omega(n\log n)$ time in the algebraic decision tree model (indeed, a special case of the problem is to compute the lower envelope of all lines, which has an $\Omega(n\log n)$ time lower bound). In particular, in the *symmetric case* where $m = n$, $\Omega(n^{4/3})$ is a lower bound since the worst case combinatorial complexity of all non-empty faces is $\Omega(n^{4/3})$ [17].

Our result. We propose a new (deterministic) algorithm of $O(m^{2/3}n^{2/3} + (m+n)\log n)$ time for the problem, which is $O(n^{4/3})$ when $m = n$ and matches the above $\Omega(n^{4/3})$ lower bound. For the *asymmetric case* where $m \neq n$, using Ben-Or's techniques [5], we prove that $\Omega(m\log n)$ is also a lower bound under the algebraic decision tree model. Combining with the above $\Omega(m^{2/3}n^{2/3} + n\log n + m)$ lower bound, we thus obtain the $\Omega(m^{2/3}n^{2/3} + (m+n)\log n)$ lower bound for solving the problem. As such, our algorithm for the asymmetric case is also optimal. Although our solution only improves the previously best result of Wang [26] by a factor of $\log^{2/3}(n/\sqrt{m})$, the result is important for providing the first optimal solution to a fundamental and classical problem in computational geometry.

Our algorithm takes a different approach than the previous work. For example, the most recent work of Wang [26] first solves the problem in $O(n\log n + m\sqrt{n\log n})$ time in the dual setting. Then, the algorithm is plugged into the framework of Agarwal [1] as a subroutine. More specifically, the framework first utilizes a cutting of L to divide the problem into a collection of subproblems and then solves these subproblems using Wang's new algorithm.

We dispense with the framework of Agarwal [1], and propose a new algorithm for the problem in the primal setting, which is different from that of Wang [26] in the dual setting. One key subproblem is to merge convex hulls, for which our techniques rely on a crucial observation that certain pairs of convex hull boundaries can intersect at most $O(1)$ times. Using a hierarchical cutting of L [8] and combining our new algorithm with Wang's, we obtain a recursive algorithm that runs in $n^{4/3} \cdot 2^{O(\log^* n)}$ time for the symmetric case $m = n$.

To further reduce the $2^{O(\log^* n)}$ factor, we resort to the recent techniques from Chan and Zheng [7]: the Γ-*algorithm framework* for bounding algebraic decision tree complexities. More specifically, after $O(1)$ recursive steps in our recursive algorithm, we reduce the problem to $O((n/b)^{4/3})$ subproblems of size $O(b)$ each (i.e., in each subproblem, we need to compute the non-empty faces of an arrangement of b lines for b points), with $b = O(\log\log n)$. As b is very small, solving the problem efficiently under the algebraic decision tree model (i.e., only count the number of comparisons) can lead to an efficient algorithm under the conventional computational model (e.g., the real RAM model). With Chan and Zheng's techniques, we developed an algorithm that solves each subproblem in $O(b^{4/3})$ decision tree complexity (i.e., the algorithm uses $O(b^{4/3})$ comparisons). Since b is small, we are able to build a decision tree for the algorithm in $O(n)$ time. With the decision tree, each subproblem can then be solved in $O(b^{4/3})$ time in the conventional computation model. Consequently, the original problem is solved in $O(n^{4/3})$ time. In addition, using this algorithm as a subroutine, the asymmetric case of the problem where $m \neq n$ can be solved in $O(m^{2/3}n^{2/3} + (m+n)\log n)$ time.

Note that our algorithm is not a direct application of Chan and Zheng's techniques. Nor it is a simple adaption of their algorithm for Hopcroft's problem. Indeed, in order to fit the Γ-algorithm framework, we need to design new procedures for a number of subproblems (notably, our techniques involve shaving log factors to merge convex hulls), which may be interesting in their own right. This is also the case for solving other problems when Chan and Zheng's techniques are applied. For instance, Chan, Cheng, and Zheng [6] recently used the technique to tackle the higher-order Voronoi diagram problem and presented an optimal

algorithm by improving the previous best algorithm by a factor of $2^{O(\log^* n)}$. To this end, they had to develop new techniques and made a great effort. On the other hand, there are other problems whose current best solutions have $2^{O(\log^* n)}$ factors, but it is not clear whether the Γ-algorithm techniques [6, 7] can be used to further reduce these factors. For instance, the biclique partition problem for a set of n points in the plane can be solved in $n^{4/3} \cdot 2^{O(\log^* n)}$ time [20, 27]. It has been open whether an $O(n^{4/3})$ time algorithm is possible.

Related work. Other related problems have also been studied in the literature, e.g., the segment case where L consists of n line segments. Although faces in an arrangement of lines are convex, they may not even be simply connected in an arrangement of segments. Therefore, the segment case becomes more challenging. It has been proved that the combinatorial complexity of all non-empty faces of the segment arrangement is bounded by $O(m^{2/3}n^{2/3} + n\alpha(n) + n \log m)$ [4] and $O(n\sqrt{m}\alpha(n))$ [13], where $\alpha(n)$ is the inverse Ackermann function; a lower bound $\Omega(m^{2/3}n^{2/3} + n\alpha(n))$ [14] was also known. To compute all non-empty faces, Edelsbrunner, Guibas, and Sharir [14] first gave a randomized algorithm of $O(m^{2/3-\delta}n^{2/3+2\delta} \log n + n\alpha(n) \log^2 n \log m)$ expected time for any $\delta > 0$. Agarwal [1] presented an improved deterministic algorithm of $O(m^{2/3}n^{2/3} \log n \log^{2.11}(n/\sqrt{m}) + n \log^3 n + m \log n)$ time. Also, Agarwal, Matoušek, and Schwarzkopf [2] derived a randomized algorithm of $O(n^{2/3}m^{2/3} \log^2 n + (n\alpha(n) + n \log m + m) \log n)$ expected time. Wang [26] proposed an $O(n^{2/3}m^{2/3} \log n + \tau(n\alpha^2(n) + n \log m + m) \log n)$ time deterministic time, where $\tau = \min\{\log m, \log(n/\sqrt{m})\}$. An intriguing question is whether our new techniques for the line case can somehow be utilized to tackle the segment case. One obstacle, for example, is that the problem for the segment case in the dual setting is not "cleanly" defined and thus we do not have a corresponding algorithm in the dual setting.

In part because of the difficulty, a special case has been studied extensively, in which we wish to compute a single face in an arrangement of segments. The problem can be solved in $O(n\alpha(n) \log n)$ expected time using a randomized algorithm [9], or in $O(n\alpha^2(n) \log n)$ time using a deterministic algorithm [3]. These algorithms provide improvements over the previous deterministic methods, which required $O(n \log^2 n)$ time [23] and $O(n\alpha(n) \log^2 n)$ time [14] respectively. An open problem in this field has been whether it is possible to have an $O(n\alpha(n) \log n)$ time deterministic algorithm. It is worth noting that computing the upper envelope of all segments can be accomplished more efficiently in $O(n \log n)$ time [19].

The online query problems have also been studied, i.e., preprocess a set of lines or segments so that the face containing a query point can be quickly computed; see, e.g., [12, 18, 26].

Outline. After introducing notation in Section 2, we present an algorithm in Section 3 that solves the problem in the primal plane. Then, in Section 4, we describe the second algorithm (which is mainly from [26]) that tackles the problem in the dual plane. We combine the two algorithms in Section 5 to obtain our final algorithm. Due to the space limit, many details and proofs are omitted but can be found in the full paper.

2 Preliminaries

We follow the same notation as in Section 1, e.g., L, P, n, m, $\mathcal{A}(L)$. Our goal is to compute all nonempty cells of $\mathcal{A}(L)$. For ease of discussion, we assume that no point of P lies on a line of L. Note that this implies that each point of P is in the interior of a face of $\mathcal{A}(L)$. We also make a general position assumption that no line of L is vertical. These assumptions can be relaxed without difficulty by the standard perturbation techniques [16, 28].

For any point p, denote by $F_p(L)$ the face of the arrangement $\mathcal{A}(L)$ that contains p. For a region R in the plane, we often use $P(R)$ to denote the subset of P in R, i.e., $P(R) = P \cap R$.

Cuttings. A tool that will be frequently used in our algorithm is cuttings [8, 22]. For a region R in the plane, we use L_R to denote the subset of lines of L that intersect the interior of R (we also say that these lines *cross* R and L_R is the *conflict list* of R).

A *cutting* for L is a collection Ξ of closed cells (each of which is a possibly unbounded triangle) with disjoint interiors, which together cover the entire plane [8, 22]. The *size* of Ξ is defined to be the number of cells in Ξ. For a parameter r with $1 \leq r \leq n$, a cutting Ξ for L is a $(1/r)$-*cutting* if $|L_\sigma| \leq n/r$ holds for every cell $\sigma \in \Xi$.

We say that a cutting Ξ' *c-refines* another cutting Ξ if each cell of Ξ' is wholly contained within a single cell of Ξ, and if every cell in Ξ encompasses at most c cells from Ξ'.

A hierarchical $(1/r)$-*cutting*, characterized by constants c and ρ, consists of a series of cuttings $\Xi_0, \Xi_1, \ldots, \Xi_k$ with the following property. Ξ_0 has a single cell that is the entire plane. For each $1 \leq i \leq k$, Ξ_i is a $(1/\rho^i)$-cutting of size $O(\rho^{2i})$ that c-refines Ξ_{i-1}. To make Ξ_k a $(1/r)$-cutting, we select $k = \Theta(\log r)$ to ensure $\rho^{k-1} < r \leq \rho^k$. Consequently, the size of Ξ_k is $O(r^2)$. As already shown in [8], it can be easily verified that the total number of cells of all cuttings Ξ_i, $0 \leq i \leq k$, is also $O(r^2)$, and the total size of the conflict lists L_σ for all cells σ of Ξ_i, $0 \leq i \leq k$, is bounded by $O(nr)$. If a cell σ within Ξ_{i-1} contains a cell σ' in Ξ_i, we call σ the *parent* of σ' and σ' a *child* of σ. In the following, we often use Ξ to denote the set of all cells of all cuttings Ξ_i, $0 \leq i \leq k$.

For any $1 \leq r \leq n$, a hierarchical $(1/r)$-cutting of size $O(r^2)$ for L (together with the conflict lists L_σ for all cells σ of Ξ_i for all $i = 0, 1, \ldots, k$) can be computed in $O(nr)$ time [8].

3 The first algorithm in the primal plane

We present our first algorithm for the problem, which works in the primal plane.

For any subset $L' \subseteq L$, let $\mathcal{U}(L')$ denote the upper envelope of the lines of L'.

Consider a point $p \in P$. Our algorithm needs to output the face $F_p(L)$ of $\mathcal{A}(L)$ that contains p. Let $L_+(p)$ (resp., $L_-(p)$) denote the subset of lines of L that are below (resp., above) p. It is not difficult to see that the face $F_p(L)$ is the common intersection of the region above the upper envelope $\mathcal{U}(L_+(p))$ and the region below the lower envelope of $L_-(p)$; see Fig. 1. Our algorithm will compute binary search trees of height $O(\log n)$ that represent $\mathcal{U}(L_+(p))$ and the lower envelope of $L_-(p)$, respectively. Using the two trees, $F_p(L)$ can be computed in $O(\log n)$ time by computing the two intersections between $\mathcal{U}(L_+(p))$ and $\mathcal{U}(L_-(p))$, and $F_p(L)$ can then be output in additional $O(|F_p(L)|)$ time. In what follows, we focus on computing $\mathcal{U}(L_+(p))$ since the lower envelope of $L_-(p)$ can be treated likewise. In the following discussion, depending on the context, an upper envelope (or lower envelope) may refer to a binary search tree that represents it. For example, the phrase "computing $\mathcal{U}(L_+(p))$" means "computing a binary search tree that represents $\mathcal{U}(L_+(p))$".

We start with computing a hierarchical $(1/r)$-cutting $\Xi_0, \Xi_1, \ldots, \Xi_k$ for the lines of L in $O(nr)$ time [8], for a parameter $r \in [1, n]$ to be determined later. Let Ξ denote the set of all cells $\sigma \in \Xi_i$, $0 \leq i \leq k$. The algorithm also produces the conflict lists L_σ for all cells $\sigma \in \Xi$. For each cell $\sigma \in \Xi_i$, $1 \leq i \leq k$, we define $\widehat{L}_\sigma$ as the subset of lines of $L_{\sigma'}$ that are completely below σ, where σ' is the parent of σ (and thus σ' is a cell of Ξ_{i-1}). As $L_{\sigma'}$ is already available and σ' has $O(1)$ children, computing $\widehat{L}_\sigma$ for all children σ of σ' can be done in $O(|L_{\sigma'}|)$ time by brute force. As such, computing $\widehat{L}_\sigma$ for all cells $\sigma \in \Xi$ takes $O(nr)$ time.

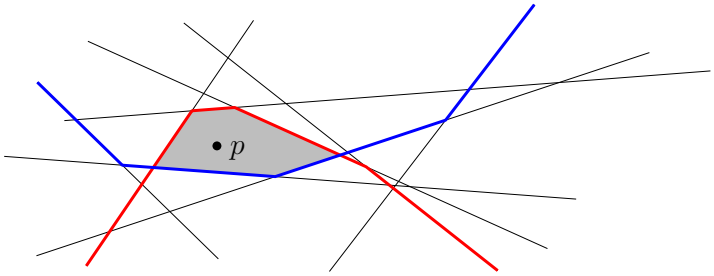


Figure 1 $F_p(L)$: the grey cell; $\mathcal{U}(L_+(p))$: blue segments; lower envelope of $L_-(p)$: red segments.

We wish to maintain a binary search tree to represent the upper envelope $\mathcal{U}(\widehat{L}_\sigma)$ of $\widehat{L}_\sigma$. To compute it in linear time, we need to have lines of $\widehat{L}_\sigma$ in sorted order by their slopes. If we sort each $\widehat{L}_\sigma$ individually, the total sorting time for all cells $\sigma \in \Xi$ is $O(nr \log n)$. To reduce the $\log n$ factor, we adopt the following strategy. We presort all lines of L in $O(n \log n)$ time. Then, we have the following lemma (see the full paper for the proof).

▶ **Lemma 1.** *Suppose we have a sorted list of the lines of L by their slopes. Then, $\widehat{L}_\sigma$ for all cells $\sigma \in \Xi$ can be sorted in $O(nr)$ time.*

After each $\widehat{L}_\sigma$ is sorted, we compute its upper envelope $\mathcal{U}(\widehat{L}_\sigma)$, which takes $O(|\widehat{L}_\sigma|)$ time. As such, computing $\mathcal{U}(\widehat{L}_\sigma)$ for all cells $\sigma \in \Xi$ can be done in $O(nr)$ time.

Next, we compute $P(\sigma)$, which is defined to be $P \cap \sigma$, for all cells $\sigma \in \Xi$. This can be done in $O(m \log r)$ time by point locations in a top-down manner in the hierarchical cutting. Specifically, for each point $p \in P$, starting from Ξ_0, which comprises a single cell that is the entire plane, suppose the cell σ of Ξ_i containing p is known; then since σ has $O(1)$ cells, locating the cell of Ξ_{i+1} containing p can be done in $O(1)$ time. As such, performing point locations for p takes $O(\log r)$ time. Note that $\sum_{\sigma \in \Xi} |P_\sigma| = O(m \log r)$ as every point of P is stored in a single cell of Ξ_i for each $0 \le i \le k$.

For each cell $\sigma \in \Xi_i$, $0 \le i \le k+1$, define $L_+(\sigma)$ as the subset of the lines of L completely below σ. We have the following lemma (see the full paper for the proof).

▶ **Lemma 2.** *For each cell $\sigma \in \Xi$, suppose σ' is the parent of σ. Then, $L_+(\sigma) = L_+(\sigma') \cup \widehat{L}_\sigma$ and thus $\mathcal{U}(L_+(\sigma))$ is the upper envelope of $\mathcal{U}(L_+(\sigma'))$ and $\mathcal{U}(\widehat{L}_\sigma)$.*

We wish to compute the upper envelope $\mathcal{U}(L_+(\sigma))$ for every cell σ in the last cutting Ξ_k. For this, we show below $\mathcal{U}(L_+(\sigma))$ for all cells $\sigma \in \Xi$ can be computed in $O(r^2 \log n)$ time.

We work on the hierarchical cutting in a top-down manner. Suppose $\mathcal{U}(L_+(\sigma'))$ for a cell σ' has been computed (which is true initially when σ' is the only cell of Ξ, in which case $L_+(\sigma') = \emptyset$ and thus $\mathcal{U}(L_+(\sigma')) = \emptyset$). Then, for each child σ of σ', we compute $\mathcal{U}(L_+(\sigma))$ as follows. By Lemma 2, $\mathcal{U}(L_+(\sigma))$ is the upper envelope of $\mathcal{U}(L_+(\sigma'))$ and $\mathcal{U}(\widehat{L}_\sigma)$. The following Lemma 3 (see the full paper for the proof) provides an algorithm that computes $\mathcal{U}(L_+(\sigma))$ based on $\mathcal{U}(L_+(\sigma'))$ and $\mathcal{U}(\widehat{L}_\sigma)$. The technical crux of the result is a crucial observation that the dual of $\mathcal{U}(L_+(\sigma'))$, which is the lower hull of the dual points of the lines of $L_+(\sigma')$, has only $O(1)$ intersections with the dual of $\mathcal{U}(\widehat{L}_\sigma)$, which is the lower hull of the dual points of the lines of $\widehat{L}_\sigma$. Note that the algorithm does not report $\mathcal{U}(L_+(\sigma))$ explicitly but rather returns a binary search tree representing $\mathcal{U}(L_+(\sigma))$, which is obtained by splitting and merging binary search trees of $\mathcal{U}(L_+(\sigma'))$ and $\mathcal{U}(\widehat{L}_\sigma)$.

▶ **Lemma 3.** *$\mathcal{U}(L_+(\sigma))$ can be obtained from $\mathcal{U}(L_+(\sigma'))$ and $\mathcal{U}(\widehat{L}_\sigma)$ in $O(\log n)$ time.*

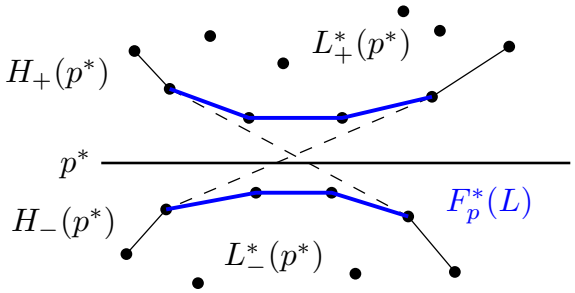


Figure 2 Illustrating the notation in the dual plane: $F_p^*(L)$, which is dual to $F_p(L)$, is composed of the blue edges between the two inner common tangents (the dashed segments).

By virtue of Lemma 3, we can compute $\mathcal{U}(L_+(\sigma))$ for all cells $\sigma \in \Xi$ in $O(|\Xi| \cdot \log n)$ time, which is $O(r^2 \log n)$ as Ξ has $O(r^2)$ cells.

Consider a cell $\sigma \in \Xi_k$. For any point $p \in P(\sigma)$, let $L_\sigma^+(p)$ denote the subset of lines of L_σ below p. It is easy to see that $L_+(p) = L_+(\sigma) \cup L_\sigma^+(p)$. Hence, $\mathcal{U}(L_+(p))$ is the lower envelope of $\mathcal{U}(L_+(\sigma))$ and $\mathcal{U}(L_\sigma^+(p))$. The above already computes $\mathcal{U}(L_+(\sigma))$. Suppose $\mathcal{U}(L_\sigma^+(p))$ is also available. Then, we compute $\mathcal{U}(L_+(p))$ by merging $\mathcal{U}(L_+(\sigma))$ and $\mathcal{U}(L_\sigma^+(p))$ in $O(\log n)$ time by the algorithm of Lemma 3. Indeed, since all lines of $L_\sigma^+(p)$ cross σ while all lines of $L_+(\sigma)$ are completely below σ, the same algorithm as Lemma 4 is also applicable here. Consequently, once the upper envelopes $\mathcal{U}(L_\sigma^+(p))$ for all points $p \in P$ are computed, $\mathcal{U}(L_+(p))$ for all $p \in P$ can be computed in additional $O(m \log n)$ time.

It remains to compute $\mathcal{U}(L_\sigma^+(p))$. To this end, we recursively apply the above algorithm on L_σ and $P(\sigma)$. A subtle issue is that while $|L_\sigma| \leq n/r$, we do not have a good upper bound for $P(\sigma)$. To address this issue, we do the following. If $|P(\sigma)| > m/r^2$, then we arbitrarily partition $P(\sigma)$ into groups of at most m/r^2 points each. Since Ξ_k has $O(r^2)$ cells, the number of groups for all cells $\sigma \in \Xi_k$ is still bounded by $O(r^2)$. Now for each cell $\sigma \in \Xi_k$, for each group P' of $P(\sigma)$, we apply the above algorithm recursively on P' and L_σ. Thus, we obtain the following recurrence for the runtime of the entire algorithm (excluding the time for presorting L), for any $1 \leq r \leq n$:

$$T(m, n) = O(nr + m \log n + r^2 \log n) + O(r^2) \cdot T(m/r^2, n/r). \tag{1}$$

We will use the recurrence later in our combined algorithm in Section 5.

4 The second algorithm in the dual plane

We discuss another algorithm, which deals with the problem in the dual plane. It mostly follows Wang's algorithm in [26] with a slight change to make the algorithm recursive.

Let P^* be the set of lines dual to the points of P and L^* the set of points dual to the lines of L. Consider a point $p \in P$. Recall that $F_p(L)$ is the face of $\mathcal{A}(L)$ that contains p. Without loss of generality, we assume that the dual line p^* of p is horizontal. In the dual plane, p^* divides the points of L^* into two subsets and the portions of the convex hulls of the two subsets between their inner common tangents are dual to $F_p(L)$ [1, 12]; see Fig 2. Specifically, define $L_+^*(p^*)$ (resp., $L_-^*(p^*)$) to be the subset of points of L^* above (resp., below) p^*. Let $H_+(p^*)$ be the lower hull of the convex hull of $L_+^*(p^*)$ and $H_-(p^*)$ the upper hull of $L_-^*(p^*)$; see Fig 2. Hence, the boundary of $F_p(L)$ is dual to the portions of $H_+(p^*)$ and $H_-(p^*)$ between their inner common tangents; let $F_p^*(L)$ denote the dual of $F_p(L)$.

The algorithm will compute binary search trees of height $O(\log n)$ to represent $H_+(p^*)$ and $H_-(p^*)$, respectively. Using the trees, their inner common tangents can be computed in $O(\log m)$ time [18] and then $F_p^*(L)$ can be reported in additional $O(|F_p^*(L)|)$ time. We only discuss how to compute $H_+(p^*)$ (i.e., compute the tree for it).

The algorithm uses a hierarchical $(1/r)$-cutting $\Xi_0, \Xi_1, \ldots, \Xi_k$ for the lines of P^*. The following recurrence for the runtime of the whole algorithm can be obtained (excluding a presorting step), for any $1 \leq r \leq m$:

$$T(m,n) = O(mr \log n + n \log r) + O(r^2) \cdot T(m/r, n/r^2). \tag{2}$$

See the full paper for the algorithm details. Recurrence (2) will be used in our combined algorithm in Section 5. We highlight some details below that are needed in Section 5.

▶ Remark. For each dual line $p^* \in P^*$ (assume it is horizontal), the algorithm for Recurrence (2) computes a set of convex hulls $\mathcal{H}_+(p^*)$ such that (1) they are completely above p^*; (2) they are pairwise disjoint and each of them is a convex hull of a subset of points of L^*; (3) $H_+(p^*)$ is the lower hull of them; (4) $\sum_{p^* \in P^*} |\mathcal{H}_+(p^*)| = O(n^{4/3})$. Again, for each convex hull of $\mathcal{H}_+(p^*)$, we have a binary search tree that represents it. As a final step, called *the convex hull merge procedure*, the algorithm computes the lower hull (which is $H_+(p^*)$) of the vertices of all convex hulls of $\mathcal{H}_+(p^*)$, for all $p^* \in P^*$. The term $O(mr \log n)$ in (2) is due to this convex hull merge procedure. If this procedure could be performed in $O(mr)$ time (which is one of our goals in Section 5), then the term in (2) would become $O(mr)$.

5 Combining the two algorithms

In this section, we combine the two algorithms presented in the last two sections to obtain a final algorithm to compute all non-empty faces of $\mathcal{A}(L)$ for the points of P.

We first discuss the symmetric case where $m = n$. If we apply (2) and then (1) using the same r, we can obtain: $T(n,n) = O(nr \log n + r^4 \log(n/r^2)) + O(r^4) \cdot T(n/r^3, n/r^3)$. Setting $r = n^{1/3}/\log n$ leads to the following

$$T(n,n) = O(n^{4/3}) + O((n/\log^3 n)^{4/3}) \cdot T(\log^3 n, \log^3 n). \tag{3}$$

The recurrence solves to $T(n,n) = n^{4/3} \cdot 2^{O(\log^* n)}$. In the following, we improve the algorithm to $O(n^{4/3})$ time using the Γ-algorithm framework of Chan and Zheng [7].

5.1 Improvement

We apply the recurrence (3) one more time to obtain the following with $b = (\log \log n)^3$:

$$T(n,n) = O(n^{4/3}) + O((n/b)^{4/3}) \cdot T(b,b). \tag{4}$$

As b is very small, we show that after $O(n)$ time preprocessing, we can solve each subproblem $T(b,b)$ of (4) in $O(b^{4/3})$ time. Consequently, we obtain $T(n,n) = O(n^{4/3})$.

More precisely, we will demonstrate that following a preprocessing step requiring $O(2^{\text{poly}(b)})$ time, where $\text{poly}(\cdot)$ represents a polynomial function, each subproblem $T(b,b)$ can be solved by performing only $O(b^{4/3})$ comparisons. Alternatively, we can solve $T(b,b)$ using an algebraic decision tree with a height of $O(b^{4/3})$. Given that $b = (\log \log n)^3$, the term $2^{\text{poly}(b)}$ remains bounded by $O(n)$. To adapt this approach to the conventional computational model (such as the standard real RAM model), we explicitly construct the decision tree for the aforementioned algorithm. This construction, which can also be viewed as part of the

preprocessing for solving $T(b, b)$, can be accomplished in $O(2^{\text{poly}(b)})$ time. Consequently, with just $O(n)$ time spent on preprocessing, we can efficiently solve each subproblem $T(b, b)$ in $O(b^{4/3})$ time. In the subsequent discussion, for notational convenience, we will use n instead of b. Our objective is to establish the following lemma.

▶ **Lemma 4.** *After $O(2^{poly(n)})$ time preprocessing, $T(n, n)$ can be solved using $O(n^{4/3})$ comparisons.*

We first apply recurrence (2) by setting $m = n$ and $r = n^{1/3}$, and obtain

$$T(n, n) = O(n^{4/3} \log n) + O(n^{2/3}) \cdot T(n^{2/3}, n^{1/3}). \tag{5}$$

As remarked at the end of Section 4, the $O(n^{4/3} \log n)$ term is due to the convex hull merge procedure for computing $H_+(p^*)$ by merging the convex hulls of $\mathcal{H}_+(p^*)$, for all $p^* \in P^*$. If the procedure could be done using $O(n^{4/3})$ comparisons, then the term would become $O(n^{4/3})$. Note that $\sum_{p^* \in P^*} |\mathcal{H}_+(p^*)| = O(n^{4/3})$. As such, to solve $T(n, n)$ by $O(n^{4/3})$ comparisons, there are two challenges: (1) perform the merge procedure on the $O(n^{4/3})$ convex hulls using $O(n^{4/3})$ comparisons; (2) solve each subproblem $T(n^{2/3}, n^{1/3})$ using $O(n^{2/3})$ "amortized" comparisons, so that all $O(n^{2/3})$ such subproblems in (5) together cost $O(n^{4/3})$ comparisons.

Γ-algorithm framework. To address these challenges, we resort to the Γ-*algorithm framework* for bounding decision tree complexities, as recently introduced by Chan and Zheng [7]. We provide a brief overview here, with more detailed information available in [7, Section 4.1].

In essence, this framework constitutes an algorithm that exclusively counts the number of comparisons, termed Γ-*comparisons* in [7], to determine if a point belongs to a semialgebraic set of degree $O(1)$ within a constant-dimensional space. Solving our problem of finding the non-empty faces is equivalent to locating the cell C^* that contains a point p^* defined by the input (i.e., the lines of L and the points of P in our problem) within an arrangement $\mathcal{A}^*$ comprising the boundaries of $O(n)$ semialgebraic sets in an $O(n)$-dimensional space (because our input size is $O(n)$). Constructing this arrangement can be achieved in $O(2^{\text{poly}(n)})$ time without inspecting the input values, thereby obviating the need for any comparisons. Notably, the number of cells in $\mathcal{A}^*$ is bounded by $n^{O(n)}$.

As the Γ-algorithm progresses, it maintains a set Π consisting of cells from $\mathcal{A}^*$. Initially, Π comprises all cells of $\mathcal{A}^*$. During the algorithm's execution, Π can only shrink, yet it always contains the sought-after cell C^*. Upon completion of the algorithm, C^* is located.

We define a potential function Φ as $\log |\Pi|$. Given that $\mathcal{A}^*$ contains $n^{O(n)}$ cells, initially Φ is $O(n \log n)$. For any operation or subroutine of the algorithm, let $\Delta\Phi$ denote the change in Φ. Since Φ monotonically decreases throughout the algorithm, we always have $\Delta\Phi \leq 0$. The sum of $-\Delta\Phi$ over the entire algorithm is $O(n \log n)$. Consequently, this allows us to accommodate costly operations or subroutines, as long as they result in a decrease in Φ.

Two algorithmic tools are introduced in [7] under the framework: the *basic search lemma* [7, Lemma 4.1] and the *search lemma* [7, Lemma A.1]. Roughly speaking, these lemmas operate as follows: when presented with a set of r predicates, where each predicate assesses whether $\gamma(x)$ holds true for the input vector x, it is guaranteed that at least one of these predicates is true for all inputs within the active cells. In such cases, the basic search lemma can identify a predicate that holds true by conducting $O(1 - r \cdot \Delta\Phi)$ comparisons. The search lemma is for scenarios involving a binary tree (or a more general directed acyclic graph of $O(1)$ degree) with nodes v, each associated with a predicate γ_v, and where for each internal node v, γ_v implies γ_u for a child u of v across all inputs in the active cells. This lemma can compute a leaf node v for which γ_v holds true with $O(1 - \Delta\Phi)$ comparisons.

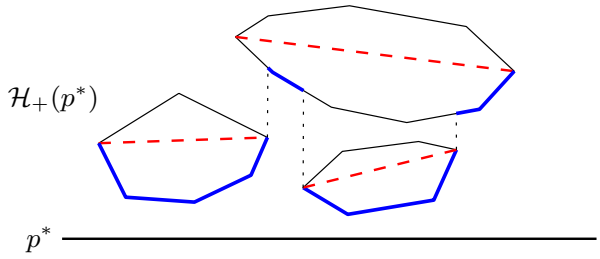


Figure 3 Illustrating the lower envelope $\mathcal{L}(\mathcal{H}_+(p^*))$ (the thick blue edges) of the convex hulls of $\mathcal{H}_+(p^*)$. The dashed segments inside convex hulls are their representative segments.

As discussed in [7], intuitively the basic search lemma provides a mild form of nondeterminism, allowing us to "guess" which one of r choices is correct, with only $O(1)$ amortized cost instead of $O(r)$. This situation naturally occurs in the context of point location, where we seek to determine which one of the r cells contains a given point. Another noteworthy application of both lemmas, as also discussed in [7], is the task of finding the predecessor of a query number within a sorted list of input numbers. As will be seen later, some subproblems in our algorithm also involves point locations as well as finding predecessors within sorted lists, making both the basic search lemma and the search lemma highly applicable.

In the following two subsections, we will address the aforementioned challenges individually. Let P represent a set of n points and L a set of n lines, for the problem in Recurrence (5).

5.2 The convex hull merge procedure

We follow the notation in the remark of Section 4. Define $K_{p^*} = |\mathcal{H}_+(p^*)|$. Hence, $\sum_{p^* \in P^*} K_{p^*} = O(n^{4/3})$. Our task is to compute (a binary search tree that represents) the lower hull (which is $H_+(p^*)$) of the vertices of all convex hulls of $\mathcal{H}_+(p^*)$, for all $p^* \in P^*$. Wang [26] gave an algorithm that can constructs $H_+(p^*)$ in $O(K_{p^*} \log n)$ time, resulting in a total of $O(n^{4/3} \log n)$ time for all $p^* \in P^*$. We will convert the algorithm to a more efficient Γ-algorithm that uses only $O(n^{4/3})$ comparisons. Below, we first review Wang's algorithm.

Constructing $H_+(p^*)$: A conventional algorithm from Wang [26]. Since $H_+(p^*)$ is the lower hull of all convex hulls of $\mathcal{H}_+(p^*)$, it is only necessary to focus on the lower hull of each convex hull of $\mathcal{H}_+(p^*)$. For each convex hull of $\mathcal{H}_+(p^*)$, since we have its binary search tree, we can obtain a binary search tree only representing its lower hull in $O(\log n)$ time. This is achieved by initially identifying the leftmost and rightmost vertices of the convex hulls and subsequently executing split/merge operations on these trees. The total time for doing this for all convex hulls of $\mathcal{H}_+(p^*)$ is $O(K_{p^*} \log n)$.

Next we compute the portions of each lower hull $H \in \mathcal{H}_+(p^*)$ that are vertically visible from the line p^*. A point $q \in H$ is *vertically visible* from p^* if the vertical segment connecting q to p^* does not intersect any other lower hull of $\mathcal{H}_+(p^*)$. Remarkably, these visible portions collectively form the lower envelope of $\mathcal{H}_+(p^*)$, denoted by $\mathcal{L}(\mathcal{H}_+(p^*))$; see Fig. 3.

For each convex hull H within $\mathcal{H}_+(p^*)$, we define its *representative segment* as the line segment connecting the leftmost and rightmost endpoints of H (see Fig. 3). Define S to be the set of the representative segments of all convex hulls of $\mathcal{H}_+(p^*)$. Because the convex hulls are pairwise disjoint, so are the segments of S. Observe that the lower envelope $\mathcal{L}(S)$ of S corresponds to $\mathcal{L}(\mathcal{H}_+(p^*))$ in the following manner. For each maximal segment $\overline{ab}$ of $\mathcal{L}(S)$, suppose it lies on the representative segment of a convex hull H of $\mathcal{H}_+(p^*)$. Then the vertical projection of $\overline{ab}$ onto the lower hull of H constitutes a maximal portion of the

lower hull of H on $\mathcal{L}(\mathcal{H}_+(p^*))$. This specific portion can be obtained in $O(\log n)$ time by splitting the binary search tree representing the lower hull of H at the x-coordinates of a and b, respectively. Consequently, once $\mathcal{L}(S)$ is available, $\mathcal{L}(\mathcal{H}_+(p^*))$, where each maximal portion is represented by a binary search tree, can be obtained in additional $O(|\mathcal{L}(S)| \log n)$ time. Since segments of S are disjoint, we have $|\mathcal{L}(S)| \leq 2K_{p^*} - 1$ and constructing $\mathcal{L}(S)$ can be accomplished in $O(K_{p^*} \log K_{p^*})$ time by a straightforward plane sweeping algorithm. Note that $K_{p^*} \leq n$ since vertices of each convex hull of $\mathcal{H}_+(p^*)$ are all from L^* (whose size is n). As such, $\mathcal{L}(\mathcal{H}_+(p^*))$ can be computed in $O(K_{p^*} \log n)$ time.

With $\mathcal{L}(\mathcal{H}_+(p^*))$, we proceed to compute the lower hull $H_+(p^*)$. As previously discussed, $\mathcal{L}(\mathcal{H}_+(p^*))$ comprises at most $2K_{p^*} - 1$ pieces, ordered from left to right. Each piece is a portion of a lower hull of $\mathcal{H}_+(p^*)$ and is represented by a binary search tree.

To begin, we merge the first two pieces by computing their lower common tangent, a task achievable in $O(\log n)$ time [24], since these two pieces are separated by a vertical line. After that we obtain a binary search tree that represents the lower hull of the first two pieces of $\mathcal{L}(\mathcal{H}_+(p^*))$. We proceed to merge this lower hull with the third piece of $\mathcal{L}(\mathcal{H}_+(p^*))$ in a similar manner. This process continues until all pieces of $\mathcal{L}(\mathcal{H}_+(p^*))$ have been merged, culminating in a binary search tree representing $H_+(p^*)$. The runtime is $O(K_{p^*} \log n)$, as each merge operation consumes $O(\log n)$ time, and $\mathcal{L}(\mathcal{H}_+(p^*))$ has at most $2K_{p^*} - 1$ pieces.

As such, the lower hull $H_+(p^*)$ can be computed in $O(K_{p^*} \log n)$ time. Applying the algorithm to all dual lines p^* of P^* will compute the lower hulls $H_+(p^*)$ for all $p^* \in P^*$. It should be noted that after the algorithm is applied to one line $p^* \in P^*$, binary search trees of convex hulls of $\mathcal{H}_+(p^*)$ may have been destroyed due to the split and merge operations during the algorithm. The destroyed convex hulls may be used later for other lines of P^*. To address the issue, we can use persistent binary search trees with path-copying [11, 25] to represent convex hulls so that standard operations on the trees (e.g., merge, split) can be performed in $O(\log n)$ time each and after each operation the original trees are still kept.

Constructing $H_+(p^*)$: a new and faster Γ-algorithm. We now convert the above algorithm to a faster Γ-algorithm. In the above algorithm, the procedures that take $O(K_{p^*} \log n)$ time are the following: (1) Computing the trees representing the lower hulls of $\mathcal{H}_+(p^*)$; (2) computing the lower envelope $\mathcal{L}(S)$ of S; (3) computing the lower envelope $\mathcal{L}(\mathcal{H}_+(p^*))$ of $\mathcal{H}_+(p^*)$ by using $\mathcal{L}(S)$; (4) computing $H_+(p^*)$ by merging the pieces of $\mathcal{L}(\mathcal{H}_+(p^*))$. Each of these procedures leads to an overall $O(n^{4/3} \log n)$ time for all $p^* \in P^*$ since $\sum_{p^* \in P^*} |\mathcal{H}_+(p^*)| = O(n^{4/3})$.

For each of these procedures, we will design a corresponding "Γ-procedure" that uses only $O(n^{4/3})$ comparisons for all $p^* \in P^*$ (thus the amortized cost for each p^* is $O(K_{p^*})$). In particular, the fourth procedure poses the most challenge and its solution is also the most interesting. See the full paper for our Γ-algorithms for the first three procedures.

5.2.1 The fourth procedure

The fourth procedure is to compute $H_+(p^*)$. The idea is to merge pieces of $\mathcal{L}(\mathcal{H}_+(p^*))$ one by one from left to right, and each merge is done by first computing the lower common tangent. Note that each piece is a portion of the lower hull of a convex hull of $\mathcal{H}_+(p^*)$ and is represented by a binary search tree. Specifically, suppose we have the lower hull H_i for the first i pieces, i.e., we have a binary search tree T_i for H_i. The next step is to merge H_i with the $(i+1)$-th piece H by first computing the lower common tangent of H_i and H. Let T_H be the binary search tree for H. With the lower common tangent, we perform split and merge operations on T_i and T_H for H to obtain a new tree T_{i+1} for the lower hull of the first $i+1$ pieces. Finding the lower common tangent can be done in $O(\log n)$ time [24] and constructing the tree T_{i+1} also takes $O(\log n)$ time.

In the following, we focus on describing a Γ-procedure that can compute the lower common tangent of H_i and H using $O(1 - n^{1/4} \cdot \Delta\Phi)$ comparisons. After the lower common tangent is computed, constructing T_{i+1} does not need any comparisons because split and merge operations on T_i and T_H do not involve any comparisons.

5.2.1.1 Notation and algorithm overview

For notational convenience, let $H_1 = H_i$, $H_2 = H$, $T_1 = T_i$, and $T_2 = T_H$.

We will use the basic search lemma of the Γ-algorithm framework of Chan and Zheng [7] discussed above. We will also rely on the technique from Overmars and van Leeuwen [24] for computing common tangents of two convex hulls.

Let the line segment $\overline{t_1t_2}$ be the lower common tangent between H_1 and H_2, with $t_1 \in H_1$ and $t_2 \in H_2$. We call t_1 and t_2 the *tangent points*. For ease of exposition, we assume that $\overline{t_1t_2}$ is not collinear with any edge of $H_1 \cup H_2$. Thus, the tangent points t_1 and t_2 are unique.

For each $i = 1, 2$, since we are searching a vertex t_i from H_i, from now on, we view H_i as a sequence of vertices ordered from left to right. We use a *chain* to refer to a contiguous subsequence of H_i. For two vertices u and v of H_i with u left of v, we use $H_i[u, v]$ to denote the chain of H_i from u to v including both u and v, while $H_i(u, v)$ is defined to be $H_i[u, v] \setminus \{u, v\}$. Similarly, $H_i[u, v)$ refers to the chain of H_i from u to v including u but excluding v; $H_i(u, v]$ is defined likewise. In addition, for any vertex u of H_i, we use $u - 1$ (resp., $u + 1$) to refer to the left (resp., right) neighboring vertex of u in H_i; for convenience, if u is the leftmost (resp., rightmost) vertex of H_i, then $u - 1$ (resp., $u + 1$) refers to u. For example, $H_i(u, v) = H_i[u + 1, v - 1]$. The following lemma is based on the technique of Overmars and van Leeuwen [24], which will be used in our algorithm.

▶ **Lemma 5** (Overmars and van Leeuwen [24]). *Given two points $p_1 \in H_1$ and $p_2 \in H_2$, p_1 (resp., p_2) partitions H_1 (resp., H_2) into two chains (we assume that neither chain contains p_i, $i = 1, 2$). Then, among the four chains of H_1 and H_2, we can determine at least one chain that does not contain a tangent point based on the following six points: p_1 (resp., p_2) and its two neighboring vertices of H_1 (resp., H_2); such a chain is called* OvL-prunable[1].

Lemma 5 provides a pruning criterion for the binary search algorithm of [24] that can compute $\overline{t_1t_2}$ in $O(\log n)$ time. We will use the lemma in a different way. For any subchain of an OvL-prunable chain determined by the lemma, we also say that the subchain is *OvL-prunable* (a subchain is a contiguous subsequence of vertices of the chain).

For each $i = 1, 2$, we maintain two vertices u_i and v_i of H_i such that $t_i \in H_i(u_i, v_i)$. After $O(1)$ iterations, either $H_i(u_1, v_1)$ or $H_i(u_2, v_2)$ contains a single point. The single point is t_1 in the former case and t_2 in the latter one. In either case, one tangent point is determined. After that, the other tangent point can be computed by a similar (and simpler) procedure. For the simplicity of discussion, we assume t_i is not an endpoint of H_i, for each $i = 1, 2$.

5.2.1.2 Computing a tangent point

Initially, for each $i = 1, 2$, we set u_i and v_i to be the two endpoints of H_i, respectively. This guarantees that the algorithm invariant holds, i.e., $t_i \in H_i(u_i, v_i)$. We consider a general step of the algorithm. To apply the basic search lemma, we define a predicate $\gamma(u_1, v_1, u_2, v_2)$ with respect to u_1, v_1, u_2, v_2, as follows.

▶ **Definition 6.** *Define $\gamma(u_1, v_1, u_2, v_2)$ to be true if neither $H_1(u_1, v_1)$ nor $H_2(u_2, v_2)$ is OvL-prunable when Lemma 5 is applied to (p_1, p_2) for any $p_1 \in \{u_1, v_1\}$ and $p_2 \in \{u_2, v_2\}$.*

[1] OvL is the last name initials of Overmars and van Leeuwen.

By Lemma 5, whether γ is true can be determined by $O(1)$ points of L^*, i.e., at most six points are needed for each pair (p_1, p_2) with $p_1 \in \{u_1, v_1\}$ and $p_2 \in \{u_2, v_2\}$, and thus the predicate is of $O(1)$ degree (where the variables are the coordinates of the $O(1)$ points).

Each iteration of the algorithm proceeds as follows. For each $i = 1, 2$, partition $H_i[u_i, v_i]$ into r chains of roughly equal lengths: $H_i[u_{ij}, v_{ij}]$, $1 \leq j \leq r$, for a parameter r to be determined later. Define $C_i = \{H_i[u_{ij} - 1, v_{ij} + 1] \mid 1 \leq j \leq r\}$. As such, the length of each chain of C_i is roughly equal to the length of $H_i[u_i, v_i]$ divided by r. Define $\mathcal{C} = C_1 \times C_2$, i.e., $\mathcal{C}$ consists of $O(r^2)$ pairs of chains such that in each pair, the first chain is from C_1 and the second one is from C_2. We have the following observation (see the full paper for the proof).

▶ **Observation 7.** *$\mathcal{C}$ must contain a pair of chains $H_1[u_{1j_1}, v_{1j_1}]$ and $H_2[u_{2j_2}, v_{2j_2}]$ such that the predicate $\gamma(u_{1j_1}, v_{1j_1}, u_{2j_2}, v_{2j_2})$ holds true.*

In light of the above observation, we apply the basic search lemma on the $O(r^2)$ cells of $\mathcal{C}$ to find a pair of chains $H_1[u_{1j_1}, v_{1j_1}]$ and $H_2[u_{2j_2}, v_{2j_2}]$ such that the predicate $\gamma(u_{1j_1}, v_{1j_1}, u_{2j_2}, v_{2j_2})$ is true. By the basic search lemma, this can be accomplished using $O(1 - r^2 \cdot \Delta\Phi)$ comparisons. The subsequent lemma (see the full paper for the proof), which proves a key property, ensures our ability to recursively employ the algorithm.

▶ **Lemma 8.** *If $\gamma(u_{1j_1}, v_{1j_1}, u_{2j_2}, v_{2j_2})$ is true, then either $t_1 \in H_1[u_{1j_1}, v_{1j_1}]$ or $t_2 \in H_2[u_{2j_2}, v_{2j_2}]$, and which case happens can be determined using $O(1)$ comparisons.*

By Lemma 8, without loss of generality, we assume $t_1 \in H_1[u_{1j_1}, v_{1j_1}]$ (and thus $t_1 \in H_1(u_{1j_1} - 1, v_{1j_1} + 1)$). This finishes the current iteration. We proceed on the next iteration with the two chains $H_1[u_{1j_1} - 1, v_{1j_1} + 1]$ and $H_2[u_2, v_2]$, i.e., update $u_1 = u_{1j_1} - 1$ and $v_1 = v_{1j_1} + 1$ (and thus $t_1 \in H_1(u_1, v_1)$). As $t_2 \in H_2(u_2, v_2)$, the algorithm invariant holds.

In this way, each iteration on $H_1[u_1, v_1]$ and $H_2[u_2, v_2]$ shrinks one of them to a subchain of size roughly $1/r$ of its original size. Since $|H_1| + |H_2| \leq n$, after $O(\log_r n)$ iterations, a tangent point can be found. The total number of comparisons is thus $O(\log_r n - r^2 \cdot \Delta\Phi)$.

The above finds one tangent point. The other one can be found by a similar but simpler algorithm, which uses $O(\log_r n - r \cdot \Delta\Phi)$ comparisons. See the full paper for details.

In summary, with $O(\log_r n - r^2 \cdot \Delta\Phi)$ comparisons, the lower tangent $\overline{t_1 t_2}$ can be computed. Setting $r = n^{1/8}$ leads to an upper bound of $O(1 - n^{1/4} \cdot \Delta\Phi)$ on the number of comparisons.

In this way, the total number of comparisons for constructing the lower hull $H_+(p^*)$ by merging the pieces of $\mathcal{L}(\mathcal{H}_+(p^*))$ is $O(K_{p^*} - n^{1/4} \cdot \Delta\Phi)$ since $\mathcal{L}(\mathcal{H}_+(p^*))$ has at most $2K_{p^*} - 1$ pieces. As $\sum_{p^* \in P^*} K_{p^*} = O(n^{4/3})$ and the sum of $-\Delta\Phi$ in the entire algorithm is $O(n \log n)$, the total number of comparisons for computing $H_+(p^*)$ for all $p^* \in P^*$ is $O(n^{4/3})$.

5.3 Solving the subproblems $T(n^{2/3}, n^{1/3})$

We can solve all $O(n^{2/3})$ subproblems $T(n^{2/3}, n^{1/3})$ in Recurrence (5) using a total of $O(n^{4/3})$ comparisons. See the full paper for details.

5.4 Putting it all together, the asymmetric case, and the lower bound

This proves Lemma 4, and thus $T(n, n)$ in Recurrence (5) can be bounded by $O(n^{4/3})$ after $O(2^{\text{poly}(n)})$ time preprocessing, as discussed before. Equivalently, $T(b, b)$ in Recurrence (4) is $O(b^{4/3})$ after $O(2^{\text{poly}(b)})$ time preprocessing. Notice that the preprocessing work is done only once and for all subproblems $T(b, b)$ in (4). Since $b = (\log \log n)^3$, we have $2^{\text{poly}(b)} = O(n)$. As such, $T(n, n)$ in (4) solves to $O(n^{4/3})$ and we thus obtain the following theorem.

▶ **Theorem 9.** *Given a set of n points and a set of n lines in the plane, we can report all faces of the line arrangement that contain at least one point in $O(n^{4/3})$ time.*

The asymmetric case is handled in the following corollary; see the full paper for the proof.

▶ **Corollary 10.** *Given a set of m points and a set of n lines in $\mathbb{R}^2$, we can report all faces of the line arrangement that contain at least one point in $O(m^{2/3}n^{2/3} + (n+m)\log n)$ time.*

Theorem 11 finally proves the lower bound below, with the help of Ben-Or's technique [5]. This justifies the optimality of Corollary 10. See the full paper for the proof.

▶ **Theorem 11.** *Computing all non-empty faces of the line arrangement requires $\Omega(m^{2/3}n^{2/3} + (n+m)\log n)$ time under the algebraic decision tree model.*

References

1. Pankaj K. Agarwal. Partitioning arrangements of lines II: Applications. *Discrete and Computational Geometry*, 5:533–573, 1990. doi:10.1007/BF02187809.
2. Pankaj K. Agarwal, Jiří Matoušek, and Otfried Schwarzkopf. Computing many faces in arrangements of lines and segments. *SIAM Journal on Computing*, 27:491–505, 1998. doi:10.1137/S009753979426616X.
3. Nancy M. Amato, Michael T. Goodrich, and Edgar A. Ramos. Computing faces in segment and simplex arrangements. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC)*, pages 672–682, 1995. doi:10.1145/225058.225285.
4. Boris Aronov, Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. The number of edges of many faces in a line segment arrangement. *Combinatorica*, 12:261–274, 1992. doi:10.1007/BF01285815.
5. Michael Ben-Or. Lower bounds for algebraic computation trees (preliminary report). In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC)*, pages 80–86, 1983. doi:10.1145/800061.808735.
6. Timothy M. Chan, Pingan Cheng, and Da Wei Zheng. An optimal algorithm for higher-order voronoi diagrams in the plane: The usefulness of nondeterminism. In *Proceedings of the 35th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4451–4463, 2024. doi:10.1137/1.9781611977912.156.
7. Timothy M. Chan and Da Wei Zheng. Hopcroft's problem, log-star shaving, 2D fractional cascading, and decision trees. In *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 190–210, 2022. doi:10.1137/1.9781611977073.10.
8. Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9(2):145–158, 1993. doi:10.1007/BF02189314.
9. Bernard Chazelle, Herbert Edelsbrunner, Leonidas J. Guibas, Micha Sharir, and Jack Snoeyink. Computing a face in an arrangement of line segments and related problems. *SIAM Journal on Computing*, 22:1286–1302, 1993. doi:10.1137/0222077.
10. Kenneth L. Clarkson, Herbert Edelsbrunner, Leonidas J. Guibas, Micha Sharir, and Emo Welzl. Combinatorial complexity bounds for arrangement of curves and spheres. *Discrete and Computational Geometry*, 5:99–160, 1990. doi:10.1007/BF02187783.
11. James R. Driscoll, Neil Sarnak, Daniel D. Sleator, and Robert E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38(1):86–124, 1989. doi:10.1016/0022-0000(89)90034-2.
12. Herbert Edelsbrunner, Leonidas J. Guibas, John Hershberger, Raimund Seidel, Micha Sharir, Jack Snoeyink, and Emo Welzl. Implicitly representing arrangements of lines or segments. *Discrete and Computational Geometry*, 4:433–466, 1989. doi:10.1007/BF02187742.
13. Herbert Edelsbrunner, Leonidas J. Guibas, János Pach, Richard Pollack, Raimund Seidel, and Micha Sharir. Arrangements of curves in the plane topology, combinatorics, and algorithms. *Theoretical Computer Science*, 92(2):319–336, 1992. doi:10.1016/0304-3975(92)90319-B.

14 Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. The complexity and construction of many faces in arrangement of lines and of segments. *Discrete and Computational Geometry*, 5:161–196, 1990. doi:10.1007/BF02187784.

15 Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986. doi:10.1137/0215023.

16 Herbert Edelsbrunner and Ernst P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9:66–104, 1990. doi:10.1145/77635.77639.

17 Herbert Edelsbrunner and Emo Welzl. On the maximal number of edges of many faces in an arrangement. *Journal of Combinatorial Theory, Series A*, 41:159–166, 1986. doi:10.1016/0097-3165(86)90078-6.

18 Leonidas J. Guibas, John Hershberger, and Jack Snoeyink. Compact interval trees: A data structure for convex hulls. *International Journal of Computational Geometry and Applications*, 1(1):1–22, 1991. doi:10.1142/S0218195991000025.

19 John Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, 33:169–174, 1989. doi:10.1016/0020-0190(89)90136-1.

20 Matthew J. Katz and Micha Sharir. An expander-based approach to geometric optimization. *SIAM Journal on Computing*, 26(5):1384–1408, 1997. doi:10.1137/S0097539794268649.

21 David G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983. doi:10.1137/0212002.

22 Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(1):157–182, 1993. doi:10.1007/BF02573972.

23 Joseph S. B. Mitchell. On computing a single face in an arrangement of line segments. Manuscript, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 1990.

24 Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2):166–204, 1981. doi:10.1016/0022-0000(81)90012-X.

25 Neil Sarnak and Robert E. Tarjan. Planar point location using persistent search trees. *Communications of the ACM*, 29:669–679, 1986. doi:10.1145/6138.6151.

26 Haitao Wang. Constructing many faces in arrangements of lines and segments. *Journal of Computational Geometry (JoCG)*, 14:287–308, 2023. doi:10.20382/jocg.v14i1a11.

27 Haitao Wang and Yiming Zhao. Improved algorithms for distance selection and related problems. *Algorithmica*, 87:908–929, 2025. doi:10.1007/s00453-025-01305-z.

28 Chee-Keng Yap. Symbolic treatment of geometric degeneracies. *Journal of Symbolic Computation*, 10:349–370, 1990. doi:10.1016/S0747-7171(08)80069-7.

Sliding Cubes in Parallel

Hugo A. Akitaya
Miner School of Computer and Information Sciences, University of Massachusetts Lowell, MA, USA

Joseph Dorfer
Institute of Algorithms and Theory, Graz University of Technology, Austria

Peter Kramer
Department of Computer Science, TU Braunschweig, Germany

Christian Rieck
Institute of Mathematics, University of Kassel, Germany

Soham Samanta
Greater Commonwealth Virtual School, Medford, MA, USA

Gabriel Shahrouzi
Miner School of Computer and Information Sciences, University of Massachusetts Lowell, MA, USA

Frederick Stock
Miner School of Computer and Information Sciences, University of Massachusetts Lowell, MA, USA

Abstract

The *sliding cubes model* serves as a well-established theoretical framework for formalizing and analyzing reconfiguration algorithms in modular robotic systems built from face-connected cubic modules. We extend the parallel sliding cubes model from two to three dimensions, presenting new algorithms, surprising complexity results, and a generalization of the best known bounds from two to three dimensions. A companion video visualizes and explains our results.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Sliding squares, parallel motion, reconfigurability, three dimensions, constant makespan, log-APX hardness, NP-hardness, worst-case optimality

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.96

Category Media Exposition

Related Version *Based on*: https://doi.org/10.48550/arXiv.2603.08537 [1]

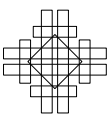
Funding *Hugo A. Akitaya, Gabriel Shahrouzi, and Frederick Stock*: Supported by the National Science Foundation (NSF), grant CCF-2348067.
Joseph Dorfer: Austrian Science Fund (FWF) 10.55776/DOC183.
Peter Kramer: Partially funded by a fellowship of the German Academic Exchange Service (DAAD) and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), grant 530918134.
Christian Rieck: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), grant 522790373.

1 Introduction

Programmable matter systems are composed of large numbers of simple modules that are capable of autonomously reconfiguring their collective shape. These systems have attracted significant attention for applications in robotics, manufacturing, and distributed computing. In practice, identifying reconfiguration sequences that transform one connected configuration into another while preserving connectivity and avoiding collisions is a highly nontrivial algorithmic problem. We consider the Sliding Cubes model introduced by Fitch, Butler,



© Hugo A. Akitaya, Joseph Dorfer, Peter Kramer, Christian Rieck, Soham Samanta, Gabriel Shahrouzi, and Frederick Stock;
licensed under Creative Commons License CC-BY 4.0



and Rus [5], where identical modules move via discrete moves in a three-dimensional lattice. Modules can perform two types of *move*, a *slide* or a *convex transition*, as shown in Figure 1. A slide translates the module by one unit along the surface of two adjacent modules, while a convex transition displaces it along the surface of only one module along two axes.

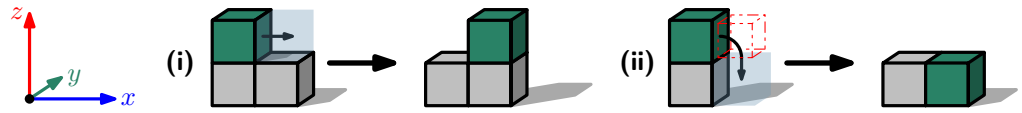


Figure 1 Two types of legal move, (i) slide and (ii) convex transition.

Our work considers the coordinated *parallel* motion of unit cubes in the three-dimensional integer grid, following and extending the notation of [2]. Parallel moves are performed in discrete *transformations* with unit-time duration; as in the two-dimensional model [2], both types of move take unit time to complete and occur at a fixed, constant speed. Transformations must preserve a connected *backbone* which requires moving modules to be *free*: moving modules do not provide connectivity. A set of modules M is free in a configuration C if the removal of any subset of M from C leaves a connected configuration. In addition, transformations must not induce *collisions*, in which the volume of two modules overlap. For instance, modules thus cannot trade places during a transformation or perform a convex transition through the same cell. We refer to [2] and Figure 2 for details.

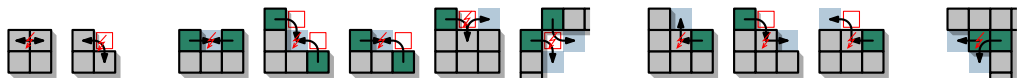


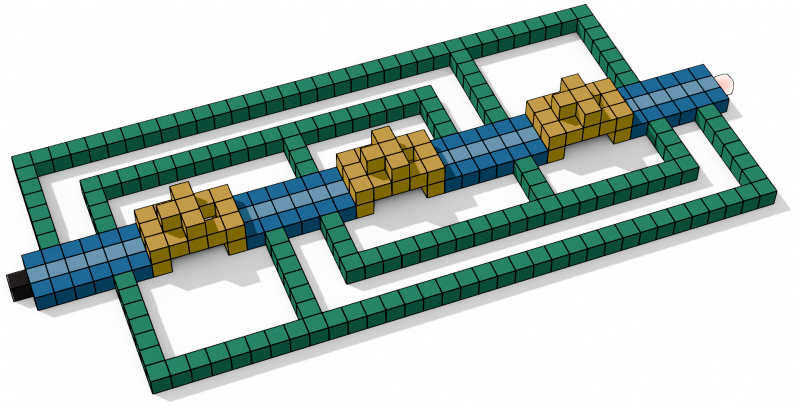
Figure 2 Examples of collisions. Figure from [2], used with the authors' permission.

2 Algorithm and complexity

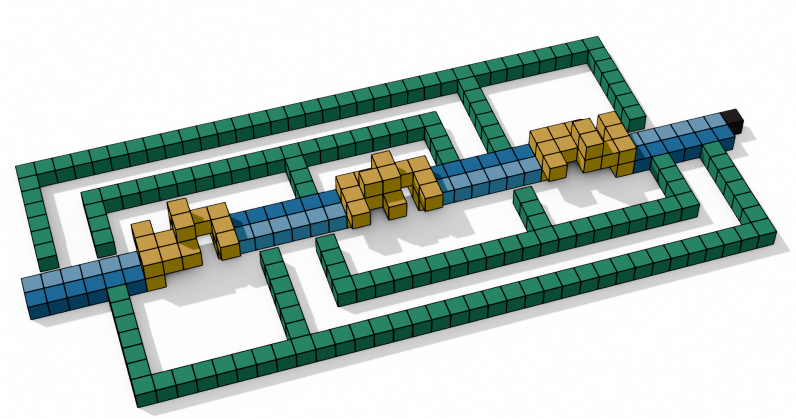
In this media exposition, we illustrate and interactively visualize three main results. All technical details of the complexity results as well as the algorithm that are presented in the accompanying video to this abstract are given in the full version [1].

1. Deciding the minimum makespan for Parallel Sliding Cubes is para-NP-hard when parameterized by the makespan and symmetric difference size: There is no polynomial time algorithm even for the case in which both values are constant (unless P = NP).
2. The makespan minimization problem in both Parallel and Sequential Sliding Cubes is log-APX-hard and cannot be approximated by a factor better than $\Theta(\log n)$.
3. Schedules for universal reconfiguration of makespan $\mathcal{O}(A + h)$ can be computed in polynomial time, where A is the area of the projection of the configuration onto the base of its bounding box and h is the height of the bounding box. In the worst case this input-sensitive bound becomes $\mathcal{O}(n)$ which is worst-case optimal.

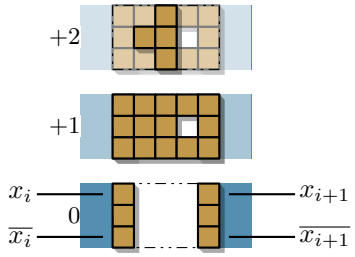
Parameterized complexity. To prove para-NP-hardness, the full paper reduces from the NP-hard problem 3Sat, which asks whether a given Boolean formula φ is satisfiable [3]. The reduction constructs two configurations, illustrated in Figure 3, using *gadgets* that encode the variables (blue) and clauses (green) of φ. These gadgets are connected by *connectors* (yellow), and the two configurations differ by a single module (black/transparent) at one end. Solving the instance in a single transformation requires a continuous path of modules to move between the two endpoints without disconnecting the structure, forcing the transformation to encode an assignment satisfying φ.



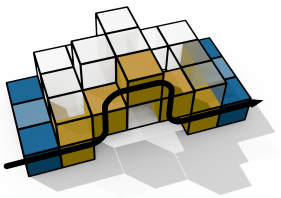
(a)



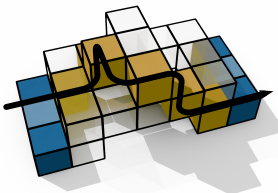
(b)



(c)



(d)



(e)

Figure 3 The construction for $\varphi = (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4})$ is shown in (a). In (b), a solution is illustrated (moving modules deleted). Variable gadgets are placed in ascending order left to right. In (c)–(e) we show the connector gadget and possible paths.

Inapproximability. The full paper shows log-APX-hardness by reduction from Set Cover [4]. An instance of this problem defines k sets $S = \{s_1, \ldots, s_k\}$ over $[n] = \{1, \ldots, n\}$ such that $\bigcup_{i=1}^{k} s_i = [n]$ and asks for a minimum cardinality subset $S^\star \subseteq S$ with $\bigcup_{s_i \in S^\star} s_i = [n]$. The construction has a bounding box of size $\mathcal{O}(n^4 k^7)$ for an instance with n elements and k sets; a schematic example is shown in Figure 4.

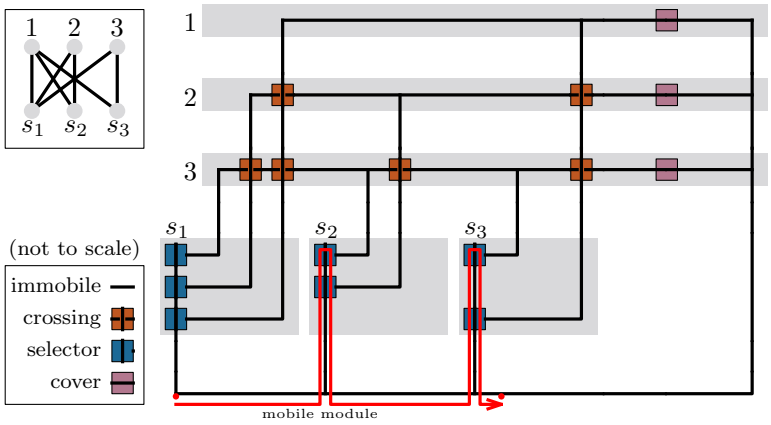


Figure 4 A diagram showing our construction for the Set Cover instance with $n = 3$ over the sets $s_1 = \{1, 2, 3\}$, $s_2 = \{2, 3\}$, and $s_3 = \{1, 3\}$. Indicated is a schedule representing $S^\star = \{s_2, s_3\}$.

A worst-case optimal algorithm. At a high level, the algorithm generalizes the methods given in [2] for two dimensions to three dimensions. The approach proceeds in three phases: (1) Gather, (2) Compress, and (3) Compact Reconfiguration. The goal of (1) Gather is to form a *snake*, which is informally a connected sequence of $5 \times 5 \times 5$ metamodules. Intuitively, the snake can move freely through the configuration without disrupting the connectivity of the remaining modules. As it moves, it collects non-cut modules, growing in size until it is sufficiently large to initiate the second phase. At a high level, in (2) Compress the snake collects the remaining modules of the configuration, creating a *compact* configuration. A configuration C is compact if for every module $\mathsf{u} \in C$, the cuboid defined by the position of u and the origin is entirely contained in the configuration:

$$\forall \mathsf{u} \in C: \quad \forall a \in [0, x(\mathsf{u})] : \forall b \in [0, y(\mathsf{u})] : \forall c \in [0, z(\mathsf{u})] : \quad (a, b, c) \in C.$$

The (3) Compact Reconfiguration phase deals with arbitrary reconfiguration of these.

3 Contributed media

To assist with developing an intuitive understanding of key pieces from the algorithm and hardness gadgets, we created visualizations of them. An interactive form of these will be hosted online using the WebVis[1] tool created by the UML Modular Robotics group and presented in [6]. We implemented new features for the tool to facilitate the creation of manual animations of reconfiguration sequences. We have compiled these implementations into an accompanying video for a presentation during CG week, similar to [7] which appeared at CG week 2024.

[1] `https://modular-robotics-group.github.io/modular-robotics/WebVis/index.html`

References

1 Hugo A. Akitaya, Joseph Dorfer, Peter Kramer, Christian Rieck, Gabriel Shahrouzi, and Frederick Stock. Sliding cubes in parallel, 2026. doi:10.48550/arXiv.2603.08537.

2 Hugo A. Akitaya, Sándor P. Fekete, Peter Kramer, Saba Molaei, Christian Rieck, Frederick Stock, and Tobias Wallner. Sliding squares in parallel. In *European Symposium on Algorithms (ESA)*, pages 28:1–28:17, 2025. doi:10.4230/LIPIcs.ESA.2025.28.

3 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *International Journal on Computational Geometry and Applications*, 22(3):187–206, 2012. doi:10.1142/S0218195912500045.

4 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing (STOC)*, pages 624–633, 2014. doi:10.1145/2591796.2591884.

5 Robert Fitch, Zack J. Butler, and Daniela L. Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2460–2467, 2003. doi:10.1109/IROS.2003.1249239.

6 UML Modular Robotics Group, Hugo A. Akitaya, Andrew Clements, Sam Downey, Jonathan Eisenbies, Soham Samanta, Gabriel Shahrouzi, and Frederick Stock. Finding shortest reconfiguration sequences for modular robots. In *Symposium on Computational Geometry (SoCG)*, pages 85:1–85:5, 2025. doi:10.4230/LIPIcs.SoCG.2025.85.

7 Irina Kostitsyna, Tim Ophelders, Irene Parada, Tom Peters, Willem Sonke, and Bettina Speckmann. Optimal in-place compaction of sliding cubes. In *Symposium on Computational Geometry (SoCG)*, pages 89:1–89:4, 2024. doi:10.4230/LIPIcs.SoCG.2024.89.

“Visualizing” the CG Community

Oswin Aichholzer
Institute of Algorithms and Theory, Graz University of Technology, Austria

Hugo A. Akitaya
Miner School of Computer and Information Sciences, University of Massachusetts Lowell, MA, USA

Anna Brötzner
Faculty of Technology and Society, Malmö University, Sweden

Peter Kramer
Department of Computer Science, TU Braunschweig, Germany

Christian Rieck
Institute of Mathematics, University of Kassel, Germany

Frederick Stock
Miner School of Computer and Information Sciences, University of Massachusetts Lowell, MA, USA

Abstract

We analyze and visualize collaboration within the Computational Geometry community by modeling co-authorship relations as a graph, where nodes correspond to individual researchers and edges represent shared publications. By aggregating and time-slicing conference data, we construct a dynamic representation of the community that supports both interactive visualization and structured search.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases CG community, visualization, graph parameters, web application

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.97

Category Media Exposition

1 Introduction

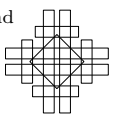
The project is motivated by an empirical observation: over the past years a member of our group repeatedly encountered co-authors of another member at different conferences where the other member was not in attendance. This prompted some speculation on our part: What is the average coauthor distance within the community? What would a co-authorship graph of the community look like? Who are the centers of this graph? What is its diameter?

We set out to study and visualize collaborative structures within the computational geometry community. In particular, we investigate the formation and persistence of cliques and larger clusters, as well as the role of individual researchers as bridges or central actors. To this end, we compute a range of graph-theoretic properties, including diameter, radius, center, and connectivity measures across different temporal snapshots, and make available through a visualization.

The idea of quantifying scholarly proximity through graph distance is well established, most prominently through the `Erdős number` [6]. The Erdős number of a researcher is defined as the length of the shortest co-authorship path to Paul Erdős, where direct co-authors have Erdős number 1, their co-authors have Erdős number 2, and so on. This concept illustrates how collaboration graphs can be used to measure relational distance and has inspired a variety of tools and datasets, such as `csauthors.net` [2], which provide searchable views of co-authorship networks in computer science, as well as the `Mathematics Genealogy`



© Oswin Aichholzer, Hugo A. Akitaya, Anna Brötzner, Peter Kramer, Christian Rieck, and Frederick Stock;
licensed under Creative Commons License CC-BY 4.0



Project [3], and the Academic Family Tree [4]. Building on these ideas, our project focuses on the computational geometry community and extends beyond static distance measures by emphasizing temporal evolution, structural metrics, and interactive exploration.

2 A web application

A variety of interactive applications exist for exploring communities and networked data, many of which are proprietary or require paid access, like PURE. In contrast, our application is tailored specifically to the CG community and is based entirely on publicly available data. This allows for open exploration of the community and its underlying structures. Moreover, another key feature of our application is the ability to explore how the community evolves over time, revealing how relationships and activity patterns develop. A snapshot of the application can be seen in Figure 1.

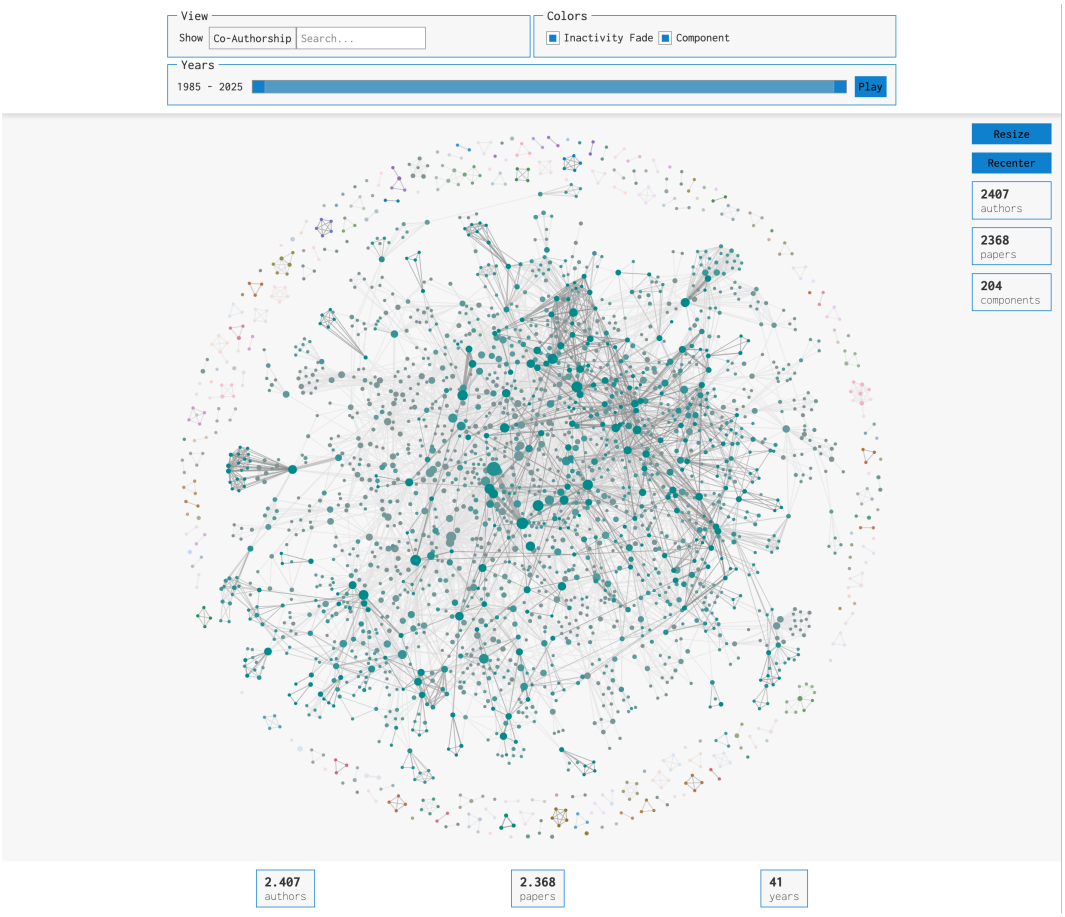


Figure 1 The web application is currently accessible via: https://cgcg.peterkramer.eu.

DBLP. Our main data source is the dblp computer science bibliography [5]. It offers bibliographic datasets covering leading journals and conference proceedings in computer science, which is updated regularly and provided as open data in the public domain. It was founded in 1993 by the University of Trier and is now operated by Schloss Dagstuhl.

Processing and visualizing the data. Upon launch, the web application retrieves the relevant data directly from `dblp` and stores it in a local `SQLite` database [7, 9] in the user's browser. Individual datapoints and their relationships are analyzed using `graphology` [8] and visualized using the `force-graph` [1] library for force-directed graph drawing.

Filtering the data. To support an interactive exploration of the CG community graph, the application offers a set of complementary filters that allow users to progressively shape and refine the view. A basic search function allows users to find specific individuals and highlight their immediate neighborhood. Temporal dynamics are conveyed through visual weighting: nodes and edges representing more recent activity are displayed with higher intensity, while older entries are shown with reduced visual prominence. In addition, node and edge sizes encode the total number of publications, and the number of joint publications.

Properties of the graph. To summarize the full story, the aggregated collaboration graph for 1985–2025 contains 2,407 researchers who collectively authored 2,368 papers, forming 204 connected components, including 82 singletons. The largest clique contains 14 researchers, whereas the average clique size is only 3.39, indicating that tightly interconnected groups remain relatively small. A similar pattern emerges when examining the connected components. Although the largest component is substantial, comprising 1,960 researchers and 2,105 papers, and having a diameter of 12 and a radius of 6, the graph is otherwise highly fragmented: the mean component size is merely 11.7 nodes, increasing to 19.05 when singleton components are excluded. This suggests that, beyond the dominant core, collaboration is dispersed across many small components. Consistently, the average number of authors per paper is relatively modest, at 2.76 overall and rising to 3.13 when single-author papers are excluded, reflecting the prevalence of small collaborative teams.

Privacy. Since all data is publicly available via `dblp`, we consider an "opt-out" policy appropriate. Authors can request removal of their bibliography from `dblp`, and our visualization, being dynamically linked, will reflect such changes. However, we do realize that someone may wish to remain listed in `dblp` but not appear in our visualization; in that case, they can contact any of the authors to have their data excluded. We also note that `dblp` treats different publication names, for example after a name change, as separate entities. We have not merged these to avoid potential privacy concerns; such merging may be requested.

3 Future work

In its current version (March 27, 2026), our web application only includes the SoCG data from `dblp`. However, it can be straightforwardly extended to include additional conferences and journals indexed by `dblp`. For example, we can readily incorporate CCCG and JoCG, for which a dataset is already available. Since EuroCG has no official proceedings, it is generally not indexed, apart from a few isolated exceptions. To capture the computational geometry community as comprehensively as possible, future work will focus on compiling a dedicated EuroCG dataset; however, this will necessarily be a gradual process, largely because not all books of abstracts are readily accessible.

In addition, we will provide ongoing support of basic features, such as allowing individuals to be anonymized if they prefer not to appear in our application, or merging multiple `dblp` identifiers corresponding to the same person, along with similar adjustments. Moreover, we will add a variety of interesting statistics over time to enrich the content. For example, whether parts of the CG community graph exhibits small-world characteristics.

Initially, we had wished to add an “affiliation” filter to visualize collaborations by institution. However, the architecture of the `dblp` database makes loading this data dynamically infeasible. We therefore leave this as a future goal.

For the CG community. This application is not intended to rank or compare individuals. Its purpose is simply to foster open and collaborative exploration and deepen our understanding of the CG community. We envision this project as a shared endeavor, “owned” and developed by our community. Therefore, we warmly invite you to explore the data, share insights, and contribute ideas or new features that will help guide its future. To actively join the project, simply reach out to any of our current members. We would love to have you on board.

References

1 Vasco Asturiano. Force-graph, force-directed graph rendered on html5 canvas. URL: `https://vasturiano.github.io/force-graph/`.
2 Raphaël P. Barazzutti. CSauthors. URL: `https://www.csauthors.net/`.
3 Harry B. Coonce. Mathematics genealogy project. URL: `https://mathgenealogy.org/index.php`.
4 Stephen David and Ben Hayden. Academic family tree. URL: `https://academictree.org/`.
5 dblp team. dblp computer science bibliography. URL: `https://dblp.org/`.
6 Jerry Grossman. The Erdös number project. URL: `https://sites.google.com/oakland.edu/grossman/home/the-erdoes-number-project`.
7 Richard D Hipp. The SQLite project. URL: `https://sqlite.org/`.
8 Guillaume Plique. Graphology, a robust and multipurpose graph object for javascript. `doi:10.5281/zenodo.5681257`.
9 Contributors to the sql.js project. sql.js.org. URL: `https://sql.js.org/`.

Interactive Uniform Floodlight Illumination and Rotating Rays Voronoi Diagrams

Carlos Alegría
FormX Inc., Sunnyvale, CA, USA

Ioannis Mantas
Planlabs, Zurich, Switzerland

Marko Savić
Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Serbia

Martin Suderland
Independent Researcher, Lugano, Switzerland

Abstract

Floodlight illumination problems are *art-gallery* variants, where a target domain needs to be illuminated by guards, each associated with a field of view. The *rotating rays Voronoi diagram* is a Voronoi diagram with rays as sites under the *angular distance*. There is a natural connection of this Voronoi structure with the problem of finding the minimum aperture such that a given set of uniform aperture floodlights illuminates a target domain. In this work we present an interactive visualization software for such problems, supporting different angular distances, namely, *oriented* and *unoriented* versions, and for different domains, namely, the plane and simple polygons.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases rotating rays Voronoi diagram, oriented angular distance, unoriented angular distance, Brocard angle, floodlight illumination, coverage problems, visualization software

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.98

Category Media Exposition

Supplementary Material *Software*: `https://github.com/VD-collective/rvd-explorer` [2]
archived at `swh:1:dir:ecbbe02e948195b997416d5a9fd082d6f6c1a01d`

Funding *Marko Savić*: Partly supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia (Grants No. 451-03-33/2026-03/ 200125 & 451-03-34/2026-03/ 200125) and by the Science Fund of the Republic of Serbia, #7462, Graphs in Space and Time: Graph Embeddings for Machine Learning in Complex Dynamical Systems – TIGRA.

1 Introduction

Given a ray r and a point $x \in \mathbb{R}^2$, the *oriented angular distance* $d_\angle(x, r)$ is the counterclockwise angle needed to rotate r around its apex until it passes through x. The *unoriented angular distance* is equal to $\min(d_\angle(x, r), 2\pi - d_\angle(x, r))$. We call α-*floodlight* the cone around r formed by the points with angular distance at most α; see Figure 1. Given a domain $\mathcal{D}$ and a set $\mathcal{S}$ of n rays, we consider the following two closely related problems.

- The **α^*-floodlight illumination** of $\mathcal{D}$ by $\mathcal{S}$: Each ray $r \in \mathcal{S}$ induces an α-floodlight and the goal is to find the minimum α, denoted by α^*, that illuminates/covers $\mathcal{D}$.
- The construction of the **rotating rays Voronoi diagram** of $\mathcal{S}$ restricted to $\mathcal{D}$, denoted by $\mathrm{RVD}(\mathcal{S})$: The diagram is a partition of $\mathcal{D}$ into regions, where each region consists of all points in $\mathcal{D}$ that are nearest to a particular ray under the angular distance function.

© Carlos Alegría, Ioannis Mantas, Marko Savić, and Martin Suderland;
licensed under Creative Commons License CC-BY 4.0



Figure 1 Oriented (left) and unoriented (right) angular distances and the respective α-floodlights.

The α^*-floodlight illumination reduces to the construction of the rotating rays Voronoi diagram, since the angle α^* is realized on a feature of $\mathrm{RVD}(\mathcal{S})$ [1]. Hence, after constructing the diagram, α^* can be found in time linear in $|\mathrm{RVD}(\mathcal{S})|$ by a traversal. Each problem has two variants, the *unoriented* and the *oriented*, depending on the distance considered.

For optimization problems where these concepts can be applied see [4, 5, 7, 12, 15, 21]. For an overview of Voronoi diagrams and illumination problems, see [6, 16] and [17, 18, 22].

Contribution. To motivate and facilitate further research on the aforementioned problems, we provide an interactive visualization software (`https://github.com/VD-collective/rvd-explorer`) that supports both angular distance functions in different domains, namely the plane and polygons. Next, we describe our software and then give an overview of results.

2 Software description

Our software RVD explorer allows users to explore rotating rays Voronoi diagrams and the α^*-floodlight illumination problem; see a screenshot of the software in Figure 2.

Users can specify a Number of sites, place their apices and directions, and select a Diagram type between RVD_RAYS_ORIENTED and RVD_RAYS_UNORIENTED. Every site, and its corresponding Voronoi region, is associated with a distinct color for clearer visualization. Show diagram skeleton highlights (in black) the graph structure of the diagram. Show point of maximum angle highlights (with a white disk) a point inside the selected frame which realizes α^*, i.e., the last point of the frame to be illuminated if all apertures started growing from 0 to α^*. The Max aperture can be bounded from above and the user can change the Current aperture (%) and also Rotate rays using the respective sliders. There are many additional features useful for studying the diagrams; the full list is shown when pressing the Help button.

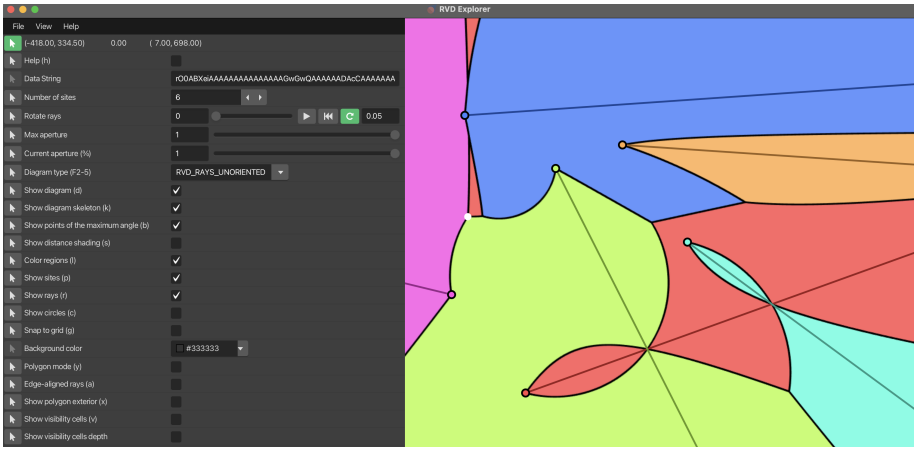


Figure 2 Image of the software showing an unoriented rotating rays Voronoi diagram in $\mathbb{R}^2$.

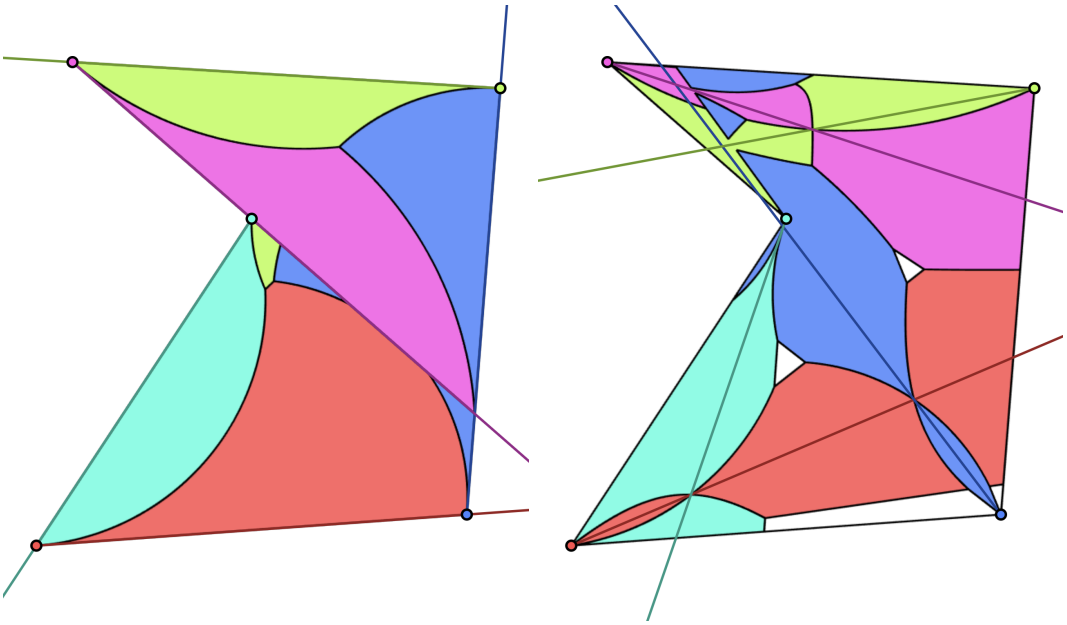


Figure 3 Diagrams of a 5-sided polygon: (left) edge-aligned, (right) $\pi/12$ bounded max aperture.

Domains. Both the entire plane and polygonal domains are supported. The domain can be chosen by toggling Polygonal domain. In the plane, rays can be placed anywhere in $\mathbb{R}^2$ and distances are evaluated over the entire $\mathbb{R}^2$; see, e.g., Figures 2 and 4. In polygonal domains, a polygon $\mathcal{P}$ is given with one ray per vertex, and distances are only evaluated inside $\mathcal{P}$. When Edge-aligned rays is enabled, each edge $\overline{v_i, v_{i+1}}$ of $\mathcal{P}$ induces a ray r_i with apex v_i passing through v_{i+1}, as in the *Brocard illumination problem* [3, 14]; see, e.g., Figures 3 and 5.

Implementation details. The application is implemented in the JavaFX graphics framework. Diagrams are rendered by pixel-based rasterization of the viewport. For each pixel, angular distances to all rays are evaluated and colored by the nearest site. Thus, for a $W \times H$ viewport, each frame renders in $O(nWH)$ time. All computations run on the CPU. To maintain a fluid interactivity, pixel rows are processed in parallel and expensive operations are avoided, e.g., angles are compared without computing their values. Diagram edges are detected by comparing neighboring pixel assignments and selectively supersampled for anti-aliasing. The user interface is built on a custom lightweight library for interactive parameter control.

3 The plane

Let the domain be $\mathbb{R}^2$. We consider both the oriented and unoriented variants; see Figure 4.

Oriented RVD($\mathcal{S}$). A comprehensive study of the oriented RVD($\mathcal{S}$) was given in [1]. A key property is that the distance function is discontinuous, at 2π, and that the bisectors consist of circular arcs and rays. The worst-case combinatorial complexity of RVD($\mathcal{S}$) is shown to be $\Omega(n^2)$ and the same is true for the complexity of a single Voronoi region. An $O(n^{2+\epsilon})$ complexity upper bound, for any $\epsilon > 0$, is obtained via 3-dimensional envelopes [20]; the same approach yields a construction algorithm running in $O(n^{2+\epsilon})$ time.

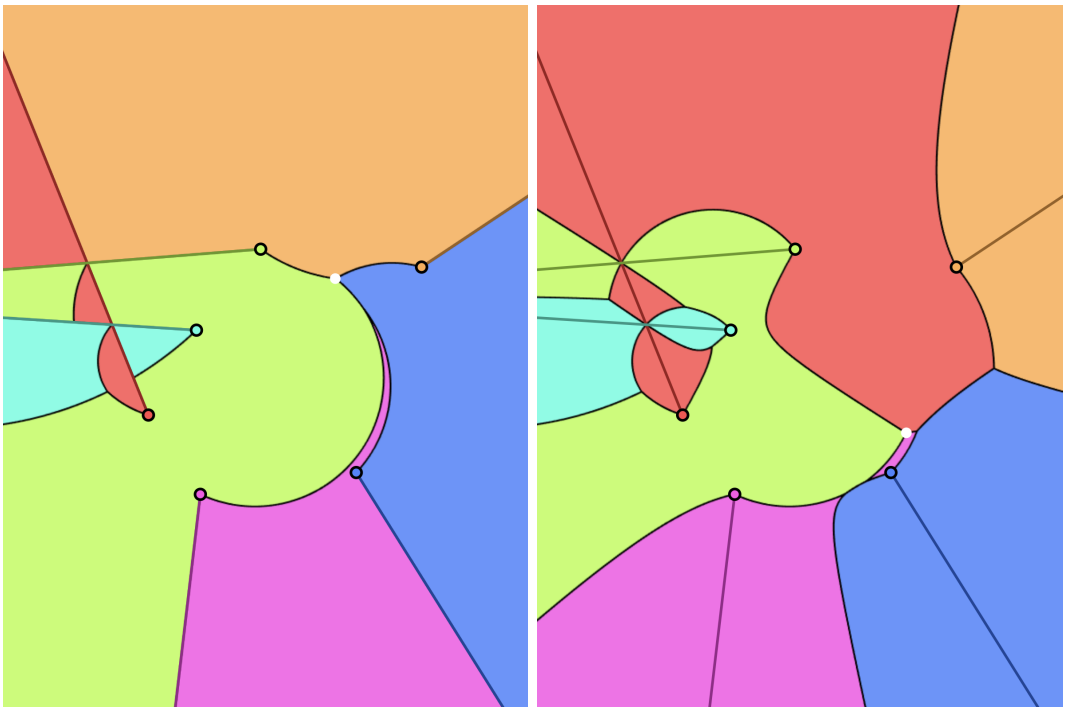


Figure 4 Voronoi diagrams of 5 rays in $\mathbb{R}^2$ in the oriented (left) and unoriented (right) variants.

Unoriented RVD($\mathcal{S}$). The unoriented RVD($\mathcal{S}$) was first proposed in [9], accompanied by an $\Omega(n^2)$ worst-case complexity bound. Its bisectors were investigated in [11]; they consist of circular and hyperbolic arcs. Notably, several results of the oriented case [1] carry over: for example, with minor adaptations of the 3-dimensional envelope methods, we can obtain an $O(n^{2+\epsilon})$ complexity upper bound and an $O(n^{2+\epsilon})$-time algorithm.

α^*-floodlight illumination. The angle α^* takes values in $[2\pi/n, 2\pi]$ in the oriented variant [1], and in the unoriented, it takes values in $[\pi/n, \pi]$. It can be obtained by a traversal after constructing RVD($\mathcal{S}$), so it requires $O(n^{2+\epsilon})$ total time [1]. It is unknown whether α^* can be computed faster directly, without RVD($\mathcal{S}$). Still, computing α^* admits an $\Omega(n \log n)$ lower bound: the simple case where all rays share the same apex is equivalent to the Max-Gap problem on a circle, which has an $\Omega(n \log n)$ lower bound in the real RAM model [13, 19].

4 Polygonal domains

We consider simple polygons as domains, with edge-aligned rays induced by the polygon boundary, i.e., related to Brocard illumination [3, 14]. For such inputs, the two variants, oriented and unoriented, coincide, as by definition only counterclockwise rotation is meaningful.

Convex polygons. The Voronoi diagram RVD($\mathcal{S}$) constrained to a convex polygon $\mathcal{P}$ was studied extensively in [1]. It is a tree, has $\Theta(n)$ combinatorial complexity, and can be constructed in optimal deterministic $\Theta(n)$ time using an elaborate divide-and-conquer scheme. This result is accompanied by a simple $O(n \log n)$-time algorithm; a similar algorithm to directly compute α^* is given in [3]. The angle α^* takes values in $(0, \pi/2 - \pi/n]$ [1].

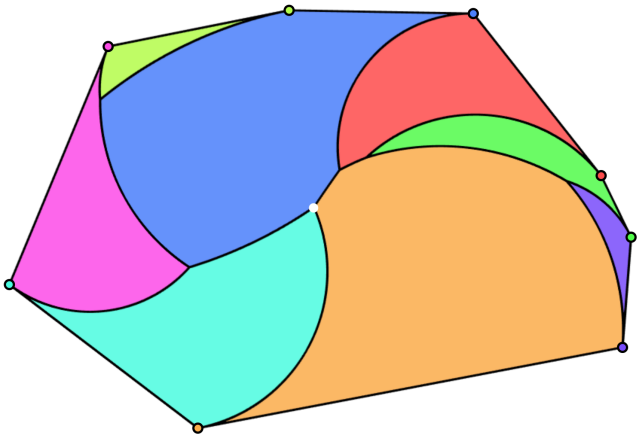


Figure 5 Voronoi diagram of a convex polygon with 8 vertices, inducing 8 edge-aligned rays.

Simple polygons. For simple polygons few results are known. The challenge comes from the presence of reflex vertices which obstruct the visibility of the rays. To construct RVD($\mathcal{S}$), an $O(n^{4+\epsilon}r)$-time algorithm was described in [14], where r is the number of reflex vertices: $\mathcal{P}$ is decomposed in $O(n^2r)$ visibility cells in $O(n^3)$ time [8, 10] and the $O(n^{2+\epsilon})$-time algorithm [1] is applied in each cell. The above implies an $O(n^{4+\epsilon}r)$ combinatorial complexity bound. A worst-case $\Omega(n^2)$ lower bound can be realized by star-shaped polygons, as the one in Figure 6.

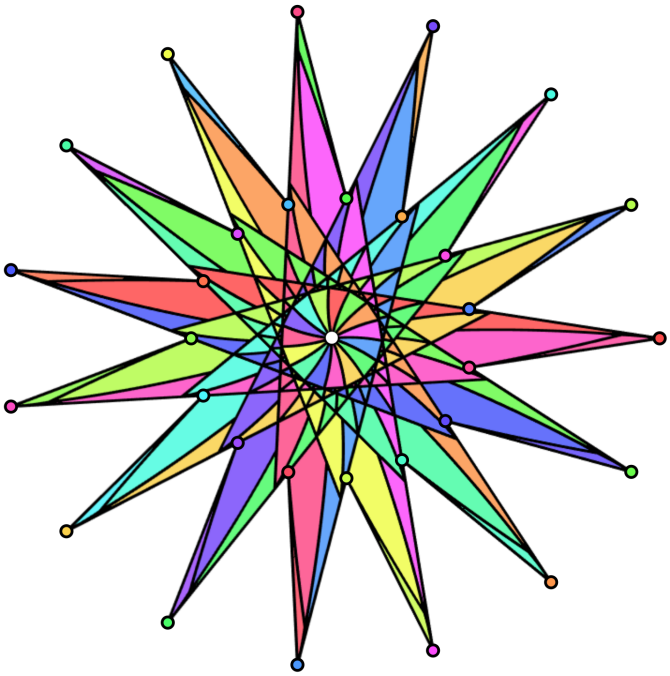


Figure 6 Diagram of a polygon with $\Theta(n^2)$ complexity, induced by the $\Theta(n^2)$ ray intersections.

Regarding the angle α^*, an $O(n^3 \log^2 n)$-time algorithm to compute it directly was given in [3]. The angle takes values in $(0, 2\pi - \pi/6)$ and instances arbitrarily close to the upper bound can be realized by windmill-shaped polygons as shown in Figure 7. It is a regular triangle with three outward wings attached on the corners.

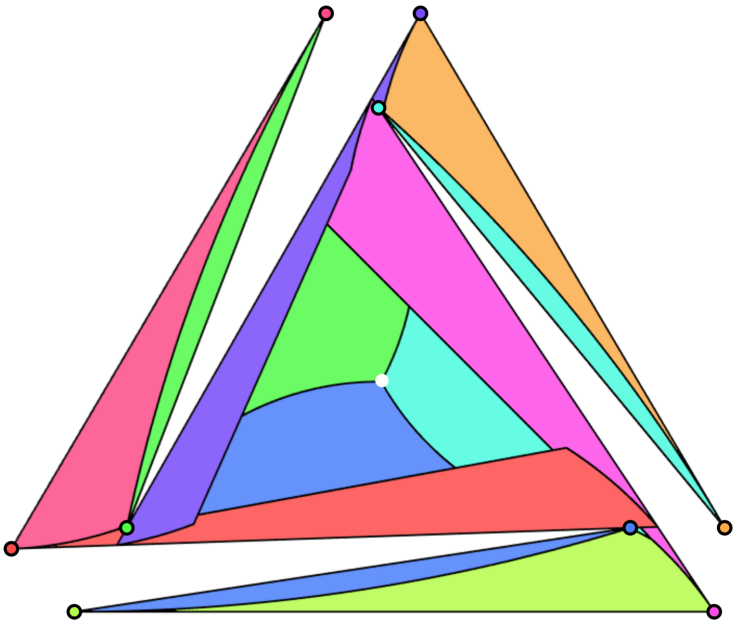


Figure 7 Voronoi diagram of a polygon with α^* close to $2\pi - \pi/6$, realized in the central triangle.

References

1 Carlos Alegría, Ioannis Mantas, Evanthia Papadopoulou, Marko Savić, Carlos Seara, and Martin Suderland. The Voronoi diagram of rotating rays with applications to floodlight illumination. *Algorithmica*, 88(2):27, 2026. doi:10.1007/s00453-025-01368-y.

2 Carlos Alegría, Ioannis Mantas, Marko Savić, and Martin Suderland. RVD-explorer. Software, version 0.1.0., swhId: swh:1:dir:ecbbe02e948195b997416d5a9fd082d6f6c1a01d (visited on 2026-05-13). URL: https://github.com/VD-collective/rvd-explorer, doi:10.4230/artifacts.26033.

3 Carlos Alegría-Galicia, David Orden, Carlos Seara, and Jorge Urrutia. Illuminating polygons by edge-aligned floodlights of uniform angle (Brocard illumination). In *Proceedings of the 33rd European Workshop on Computational Geometry (EuroCG 2017)*, pages 281–284, 2017.

4 Omur Arslan, Hancheng Min, and Daniel E. Koditschek. Voronoi-based coverage control of pan/tilt/zoom camera networks. In *2018 IEEE International Conference on Robotics and Automation (ICRA 2018)*, pages 5062–5069. IEEE, 2018. doi:10.1109/icra.2018.8460701.

5 Tetsuo Asano, Hisao Tamaki, Naoki Katoh, and Takeshi Tokuyama. Angular Voronoi diagram with applications. In *2006 3rd International Symposium on Voronoi Diagrams in Science and Engineering*, pages 18–24. IEEE, 2006. doi:10.1109/ISVD.2006.9.

6 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013. doi:10.1142/8685.

7 Piotr Berman, Jieun Jeong, Shiva P. Kasiviswanathan, and Bhuvan Urgaonkar. Packing to angles and sectors. In *Proceedings of the 19th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'07)*, pages 171–180, 2007. doi:10.1145/1248377.1248405.

8 Prosenjit Bose, Anna Lubiw, and J. Ian Munro. Efficient visibility queries in simple polygons. *Computational Geometry*, 23(3):313–335, 2002. doi:10.1016/s0925-7721(01)00070-0.

9 Mark de Berg, Joachim Gudmundsson, Herman Haverkort, and Michael Horton. Voronoi diagrams with rotational distance costs. In *Abstracts of the Computational Geometry Week: Young Researchers Forum (CG:YRF)*, 2017.

10 Leonidas J Guibas, Rajeev Motwani, and Prabhakar Raghavan. The robot localization problem. *SIAM Journal on Computing*, 26(4):1120–1138, 1997. doi:10.1137/S0097539792233257.

11 Herman Haverkort and Rolf Klein. Hyperbolae are the locus of constant angle difference. *arXiv preprint arXiv:2112.00454*, 2021. arXiv:2112.00454.

12 Evangelos Kranakis, Danny Krizanc, and Oscar Morales. Maintaining connectivity in sensor networks using directional antennae. In *Theoretical Aspects of Distributed Computing in Sensor Networks*, pages 59–84. Springer, 2011. doi:10.1007/978-3-642-14849-1_3.

13 Der-Tsai Lee and Ying-Fung Wu. Geometric complexity of some location problems. *Algorithmica*, 1(1):193–211, 1986. doi:10.1007/BF01840442.

14 Ioannis Mantas. *Problems on Planar Voronoi Diagrams*. PhD thesis, Università della Svizzera Italiana, 2022.

15 Azin Neishaboori, Ahmed Saeed, Khaled A. Harras, and Amr Mohamed. On target coverage in mobile visual sensor networks. In *Proceedings of the 12th ACM International Symposium on Mobility Management and Wireless Access (MobiWac'15)*, pages 39–46, 2014. doi:10.1145/2642668.2642671.

16 Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. John Wiley & Sons, 2009.

17 Joseph O'Rourke. *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford, 1987.

18 Joseph O'Rourke. Visibility. In *Handbook of Discrete and Computational Geometry*, pages 875–896. CRC Press, 2017.

19 Vera Sacristán. Lower bounds for some geometric problems. Technical report, Technical Report MA2-IR-98, 1998.

20 Micha Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. *Discrete & Computational Geometry*, 12(3):327–345, 1994. doi:10.1007/BF02574384.

21 Tsuyoshi Taki, Jun-ichi Hasegawa, and Teruo Fukumura. Development of motion analysis system for quantitative evaluation of teamwork in soccer games. In *Proceedings of the 3rd IEEE International Conference on Image Processing*, volume 3, pages 815–818. IEEE, 1996. doi:10.1109/ICIP.1996.560865.

22 Jorge Urrutia. Art gallery and illumination problems. In *Handbook of Computational Geometry*, pages 973–1027. Elsevier, 2000. doi:10.1016/B978-044482537-7/50023-1.

Proximity Alert: Ipelets for Neighborhood Graphs and Clustering

Gitan Balogh
Department of Computer Science,
University of Maryland, College Park, MD, USA

June Cagan
Department of Computer Science,
University of Maryland, College Park, MD, USA

Bea Fatima
Department of Physics, Utrecht University,
The Netherlands

Auguste H. Gezalyan
Université de Lorraine, CNRS, Inria, LORIA,
F-54000 Nancy, France

Danesh Sivakumar
Department of Computer Science,
University of Maryland, College Park, MD, USA

Arushi Srinivasan
Department of Computer Science,
University of Maryland, College Park, MD, USA

Yixuan Sun
Department of Computer Science,
University of Maryland, College Park, MD, USA

Vahe Zaprosyan
Department of Computer Science,
University of Maryland, College Park, MD, USA

David M. Mount
Department of Computer Science,
University of Maryland, College Park, MD, USA

Abstract

Neighborhood graphs and clustering algorithms are fundamental structures in both computational geometry and data analysis. Visualizing them can help build insight into their behavior and properties. The Ipe extensible drawing editor, developed by Otfried Cheong, is a widely used software system for generating figures. One particular aspect of Ipe is the ability to add Ipelets, which extend its functionality. Here we showcase a set of Ipelets designed to help visualize neighborhood graphs and clustering algorithms. These include: ε-neighbor graphs, furthest-neighbor graphs, Gabriel graphs, k-nearest neighbor graphs, k^{th}-nearest neighbor graphs, k-mutual neighbor graphs, k^{th}-mutual neighbor graphs, asymmetric k-nearest neighbor graphs, asymmetric k^{th}-nearest neighbor graphs, relative-neighbor graphs, sphere-of-influence graphs, Urquhart graphs, Yao graphs, and clustering algorithms including complete-linkage, DBSCAN, HDBSCAN, k-means, k-means++, k-medoids, mean shift, and single-linkage. Our Ipelets are all programmed in Lua and are freely available.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases neighborhood graphs, clustering, proximity graphs, Ipelets, visualization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.99

Category Media Exposition

Supplementary Material
Software (Source Code): `https://github.com/Otcavo/Ipelets-Full-Library` [1]
Audiovisual (Video): `https://youtu.be/NQr-SY1x2qc` [2]

1 Introduction

We present a set of **Ipelets** for visualizing clustering and neighborhood graph algorithms on point sets in $\mathbb{R}^2$. These Ipelets include: ε-neighbor graphs, furthest-neighbor graphs, Gabriel graphs, k-nearest neighbor graphs, k^{th}-nearest neighbor graphs, k-mutual neighbor graphs, k^{th}-mutual neighbor graphs, asymmetric k-nearest neighbor graphs, asymmetric k^{th}-nearest neighbor graphs, relative-neighbor graphs, sphere-of-influence graphs, Urquhart graphs, Yao graphs, and clustering algorithms including complete-linkage, DBSCAN, HDBSCAN,

© Gitan Balogh, June Cagan, Bea Fatima, Auguste H. Gezalyan, Danesh Sivakumar, Arushi Srinivasan, Yixuan Sun, Vahe Zaprosyan, and David M. Mount;
licensed under Creative Commons License CC-BY 4.0

k-means, k-means++, k-medoids, mean shift, and single-linkage. Some of these are also implemented in CGAL [9]. All our Ipelets are programmed in Lua and are freely available on GitHub. To install any Ipelet, download the file from the GitHub repository, and place it in the `ipelets` sub-folder of your Ipe folder. This work builds on previous ipelets to add functionality in various contexts [4, 8, 12, 14]. Our contribution is in line with that of several other researchers who have worked on neighborhoods of point sets, such as Voronoi diagrams and Delaunay triangulations [4, 14], and β-skeletons [11]. Ipelets can be invoked from the "Ipelets" menu, see Figure 1 for usage instructions.

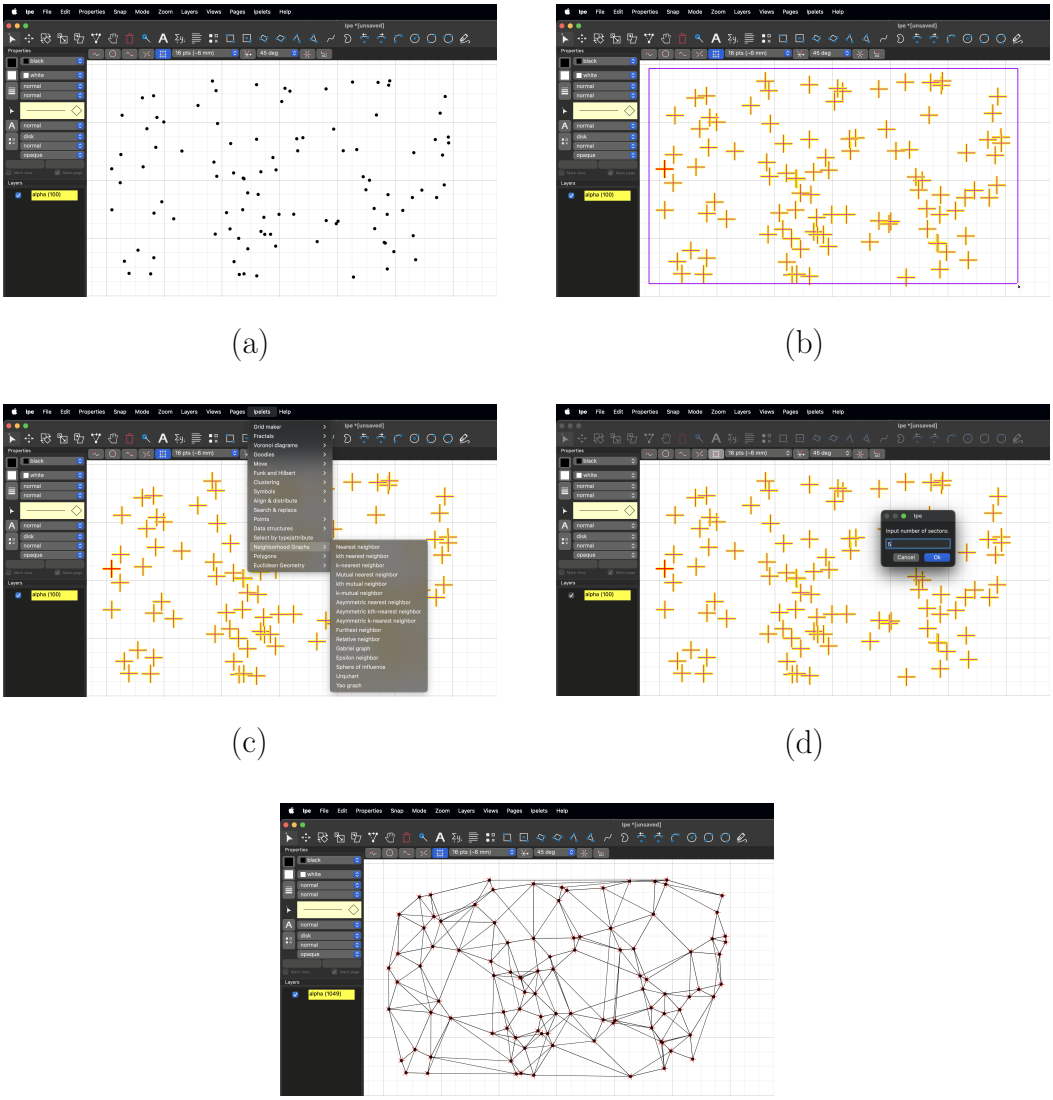


Figure 1 (a) Place a point set in Ipe, (b) select the points, (c) choose a graph from the "Ipelets" menu, (d) enter a parameter value if prompted, (e) the resulting graph is drawn automatically.

2 Neighborhood graphs

In this section, we describe the neighborhood graphs our Ipelets compute on point sets. We loosely separate them by type and briefly describe our contributions.

2.1 Nearest-neighbor variants

Nearest neighborhood graphs are fundamental tools in data analysis on a point set P. A point $p \in P$ is connected to $q \in P$ when one is the nearest neighbor of the other. The *k-nearest neighbor graph* connects p to q when q is among the k nearest neighbors of p; the *k^{th}-nearest neighbor graph* connects p to q when q is the k^{th} nearest neighbor of p; *mutual nearest neighbor graphs* connect p to q when each is the other's nearest neighbor; the *k-mutual neighbor graph* connects p to q when each is among the other's k nearest neighbors; the *k^{th}-mutual neighbor graph* connects p to q when each is exactly the other's k^{th} nearest neighbor; *asymmetric nearest neighbor graphs* connect p to q when exactly one is the other's nearest neighbor; *asymmetric k-nearest neighbor graphs* connect p to q when exactly one has the other among their k nearest neighbors; *asymmetric k^{th}-nearest neighbor graphs* connect p to q when exactly one has the other as their k^{th} nearest neighbor; and *furthest neighbor graphs* connect p to q when q is p's furthest neighbor. See illustration in Figure 2.

2.2 Proximity graphs

A proximity graph is a graph in which points are connected when a region-based condition is satisfied, often in the context of network analysis. The *Gabriel graph* connects two points p and q when the disk defined by the diameter of p and q contains no other points [10]. A *relative neighborhood graph* connects two points p and q exactly when no point is closer to both p and q than they are to each other, or equivalently, no point is in the lune of p and q (intersection of the ball at p with q on its boundary and the ball at q with p on its boundary) [15]. A *sphere of influence graph* assigns to each point a disk with radius given as the distance to its nearest neighbor and connects points when their disks overlap [16]. An *ε-graph* connects pairs of points if they lie within ε of each other. A *Urquhart graph* is the Delaunay triangulation with the longest edge of each triangle removed [17]. A *Yao graph* divides the plane into sectors by angle around points, and points are connected to their closest neighbor per sector [18]. See visuals in Figure 3.

3 Clustering

In this section, we describe the clustering algorithms we implemented for point sets. As before, we loosely separate them by type and briefly describe our contributions.

3.1 Centroid-based

Centroid-based clustering partitions a point set in space into clusters using a centroid-based method. The *k-Means* clustering partitions points into k clusters, assigning points to its nearest centroid and recomputing the centroids iteratively until it is stable. The *k-Means++* variant is similar, except it chooses initial centroids that are likely to be far apart to improve convergence. Unlike *k-means*, *k-Medoids* clustering uses preexisting points in the data as the centroids of the cluster [13]. Each iteration, a point replaces the medoid if it decreases the total intra-cluster distance. Illustrations of the methods can be seen in Figure 4.

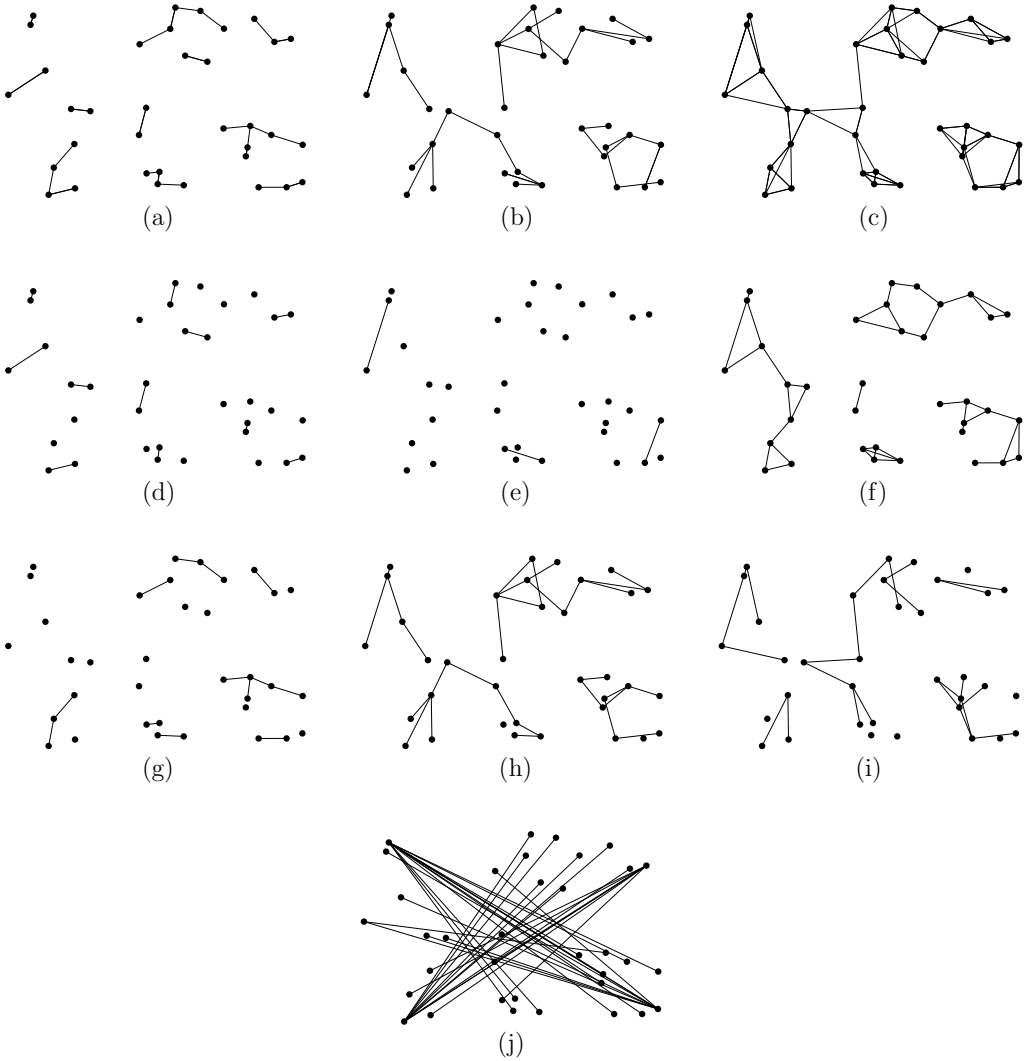


Figure 2 (a) Nearest, (b) k^{th}-nearest, (c) k-nearest, (d) mutual nearest, (e) k^{th}-mutual, (f) k-mutual, (g) asymmetric nearest, (h) asymmetric k^{th}-nearest, (i) asymmetric k-nearest, and (j) furthest neighbor graphs. In each $k = 3$.

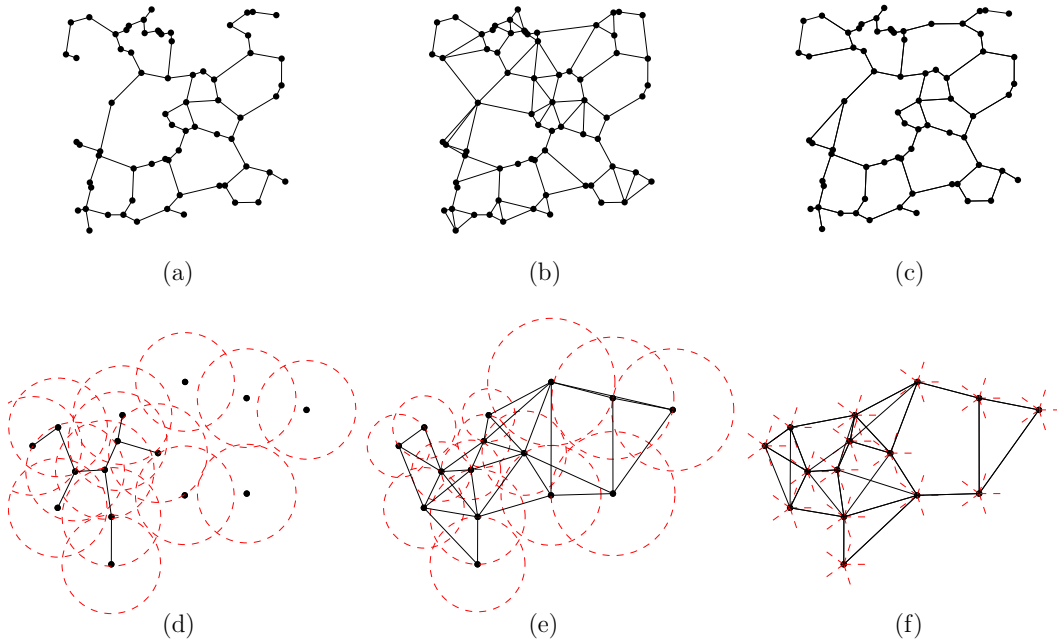


Figure 3 (a) Relative neighbor, (b) Gabriel graph, (c) Urquhart graph, (d) ε-neighbor ($\varepsilon = 28$), (e) sphere of influence, and (f) Yao graph ($k = 5$).

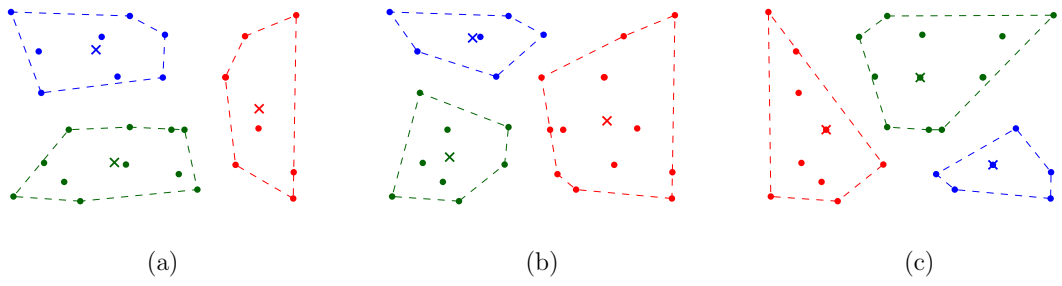
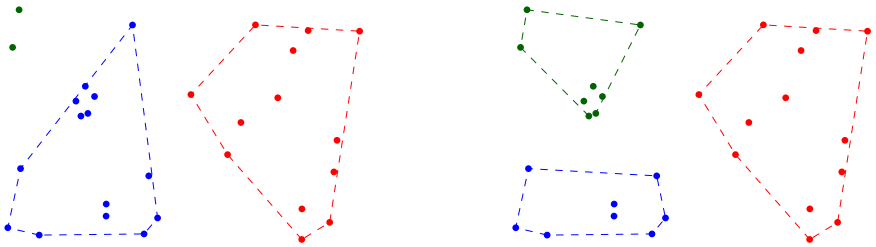


Figure 4 (a) k-Means, (b) k-Means++, and (c) k-Medoids.

3.2 Hierarchical

Hierarchical clustering methods work by treating each point as a cluster and iteratively merging clusters until only a specified number remain. *Single-linkage clustering* defines the cluster distance between clusters as the minimum pairwise distance between their points, while *complete-linkage clustering* uses the maximum [7]. See Figure 5.

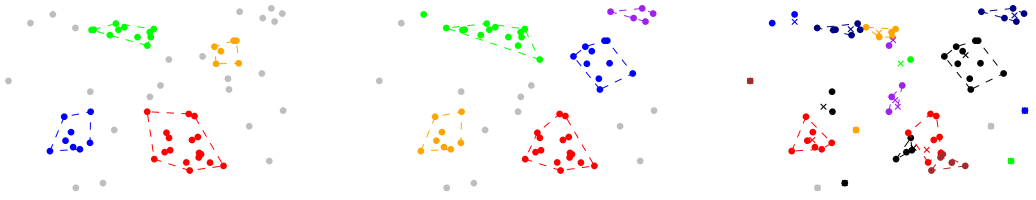


(a) (b)

Figure 5 (a) Single Linkage Clustering, (b) Complete Linkage Clustering.

3.3 Density-based

Density-based clustering partitions a point set into clusters based on how densely points are packed. *DBSCAN* groups points that are within a specified distance, ε, of each other and have a minimum number of neighboring points (MinPts). Points meeting these criteria form clusters (distinguished by colors), while points that don't are labeled as noise [6]. *HDBSCAN*, on the other hand, applies a hierarchical approach to density-based clustering [3]. It builds a mutual reachability graph and a minimum spanning tree to identify clusters at varying densities. Instead of relying on a fixed distance threshold like *DBSCAN*, *HDBSCAN* evaluates the stability of clusters across different scales. It then uses the Excess of Mass method to select stable clusters. *Mean shift* computes where each point would converge if moved towards the mean of its neighbors within the given bandwidth radius. Points whose trajectories converge on the same location form a cluster [5]. See Figure 6.



(a) (b) (c)

Figure 6 (a) HDBSCAN, (b) DBSCAN, and (c) Mean shift.

4 Conclusion

In this paper, we presented a collection of Ipelets for visualizing neighborhood graphs and clustering algorithms on point sets in $\mathbb{R}^2$. These tools are designed to assist in the understanding of these structures and their properties, and we hope they will be useful to researchers, educators, and artists alike. All Ipelets are freely available on GitHub.

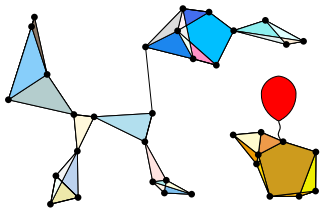


Figure 7 Artwork made using the k nearest neighbors Ipelet.

References

1 Gitan Balogh, June Cagan, Bea Fatima, Auguste H. Gezalyan, Danesh Sivakumar, Arushi Srinivasan, Yixuan Sun, Vahe Zaprosyan, and David M. Mount. Proximity Alert: Ipelets for Neighborhood Graphs and Clustering. Software (visited on 2026-05-12). URL: https://github.com/Otcavo/Ipelets-Full-Library, doi:10.4230/artifacts.25979.

2 Gitan Balogh, June Cagan, Bea Fatima, Auguste H. Gezalyan, Danesh Sivakumar, Arushi Srinivasan, Yixuan Sun, Vahe Zaprosyan, and David M. Mount. Proximity Alert: Ipelets for Neighborhood Graphs and Clustering. Audiovisual (visited on 2026-05-12). URL: https://youtu.be/NQr-SY1x2qc, doi:10.4230/artifacts.25983.

3 Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining*, pages 160–172. Springer, 2013. doi:10.1007/978-3-642-37456-2_14.

4 Otfried Cheong. The IPE extensible drawing editor. Version 7.2.28, 2024-03-12. URL: https://ipe.otfried.org/.

5 Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. doi:10.1109/34.1000236.

6 Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996. URL: http://www.aaai.org/Library/KDD/1996/kdd96-037.php.

7 Brian S. Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. *Cluster Analysis*. Wiley, 5th edition, 2011. doi:10.1002/9780470977811.

8 Klint Faber, Auguste H. Gezalyan, Adam Martinson, Aniruddh Mutnuru, Nithin Parepally, Ryan Parker, Mihil Sreenilayam, Aram Zaprosyan, and David M. Mount. French onion soup, ipelets for points and polygons (media exposition). In *41st International Symposium on Computational Geometry (SoCG 2025)*, pages 83:1–83:6. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SoCG.2025.83.

9 Andreas Fabri and Sylvain Pion. CGAL: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 538–539, 2009. doi:10.1145/1653771.1653865.

10 K. Ruben Gabriel and Robert R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259–278, 1969. doi:10.2307/2412323.

11 David G. Kirkpatrick and John D. Radke. A framework for computational morphology. In G. T. Toussaint, editor, *Computational Geometry*, pages 217–248. North-Holland, 1985. doi:10.1016/B978-0-444-87806-9.50013-X.

12 Nithin Parepally, Ainesh Chatterjee, Auguste H. Gezalyan, Hongyang Du, Sukrit Mangla, Kenny Wu, Sarah Hwang, and David M. Mount. Ipelets for the convex polygonal geometry (media exposition). In *40th International Symposium on Computational Geometry (SoCG 2024)*, pages 92:1–92:7, 2024. doi:10.4230/LIPIcs.SoCG.2024.92.

13 Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, 2009. doi:10.1016/j.eswa.2008.01.039.

14 postechDNN. Ipelet. https://github.com/postechDNN/postechDNN/commits/ipelet/dnn/IPELET, 2022. (Commit hash: de82a8d).

15 Godfried T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980. doi:10.1016/0031-3203(80)90066-7.

16 Godfried T. Toussaint. A graph-theoretical primal sketch. *Computational Morphology*, pages 229–260, 1988.

17 RB Urquhart. Algorithms for computation of relative neighbourhood graph. *Electronics Letters*, 16(14):556–557, 1980.

18 Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982. doi:10.1137/0211059.

Visualizing Higher Order Structures, Overlap Regions, and Clustering in the Hilbert Geometry

Hridhaan Banerjee
Thomas Jefferson High School for Science and Technology, Alexandria, VA, USA

Soren Brown
Department of Computer Science, University of Maryland, College Park, MD, USA

June Cagan
Department of Computer Science, University of Maryland, College Park, MD, USA

Auguste H. Gezalyan
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Megan Hunleth
Montgomery Blair High School, Silver Spring, MD, USA

Veena Kailad
Montgomery Blair High School, Silver Spring, MD, USA

Chaewoon Kyoung
Montgomery Blair High School, Silver Spring, MD, USA

Rowan Shigeno
Department of Mathematics, Haverford College, PA, USA

Yasmine Tajeddin
Department of Computer Science, University of Maryland, College Park, MD, USA

Andrew Wagger
Department of Computer Science, University of Maryland, College Park, MD, USA

Kelin Zhu
Department of Mathematics, University of Maryland, College Park, MD, USA

David M. Mount
Department of Computer Science, University of Maryland, College Park, MD, USA

Abstract

Higher-order Voronoi diagrams and Delaunay mosaics in polygonal metrics have only recently been studied, yet no tools exist for visualizing them. We introduce a tool that fills this gap, providing dynamic interactive software for visualizing higher-order Voronoi diagrams and Delaunay mosaics along with clustering and tools for exploring overlap and outer regions in the Hilbert polygonal metric. We prove that k^{th} order Voronoi cells are not always star-shaped and establish complexity bounds for our algorithm, which generates all order Voronoi diagrams at once. Our software unifies and extends previous tools for visualizing the Hilbert, Funk, and Thompson geometries.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Hilbert metric, Funk metric, Voronoi diagrams

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.100

Category Media Exposition

Supplementary Material *Software (Web Application)*: `https://andrew.wagger.net/hilbert/` [3]
Audiovisual (Video): `https://youtu.be/yhJn--3Qgks` [4]

Funding REU-CAAR 2025

1 Introduction

The Hilbert metric has attracted recent interest due to its applications in clustering [17], non-linear embeddings [18], convex approximation [1, 2], and real analysis [14]. As such, there has been work on extending results from computational geometry to the Hilbert metric, such as Voronoi diagrams (including farthest-point Voronoi) [9, 12, 15], minimum enclosing balls [6], and Delaunay triangulations [11]. To aid in these endeavors, software to help visualize the Hilbert metric has been progressively developed. This began with a Hilbert ball

© Hridhaan Banerjee, Soren Brown, June Cagan, Auguste H. Gezalyan, Megan Hunleth, Veena Kailad, Chaewoon Kyoung, Rowan Shigeno, Yasmine Tajeddin, Andrew Wagger, Kelin Zhu, and David M. Mount;
licensed under Creative Commons License CC-BY 4.0

visualizer [16], then some Voronoi demo software [9], which evolved into software for the Funk and Thomson polygonal geometry [5]. We build on this foundation by providing the first software with interactive support for dynamically visualizing higher-order Voronoi diagrams and Delaunay mosaics in the Hilbert metric, exploring overlap and outer regions (defined in Section 3), and step-by-step clustering on arbitrary convex polygonal domains. The software is implemented in JavaScript, it runs interactively in the browser, and it includes a tutorial video and brute force Voronoi mode for verification. It also maintains the features from all previous software.

2 Higher order structures

The study of k^{th} order abstract Voronoi diagrams started with "On the complexity of higher order abstract Voronoi diagrams" where cells were determined by the k closest sites [7]. The Hilbert metric does not immediately fall into the category of k^{th} order abstract Voronoi diagrams, but can be made to fit it by extending bisectors to infinity without the extensions crossing. As such, there exist frameworks for efficient k^{th} order Voronoi diagrams [7, 8]. Our visualization generates all orders at once using an edge labeling technique similar to those used in Euclidean geometry [10]. For completeness, we begin this section by proving two supporting lemmas that show our algorithm works. Then we present the algorithm. Throughout, we assume that we have been given some convex polygonal region Ω with m sides and a set of n sites S. Both proofs use a classical technique using properties of balls [13].

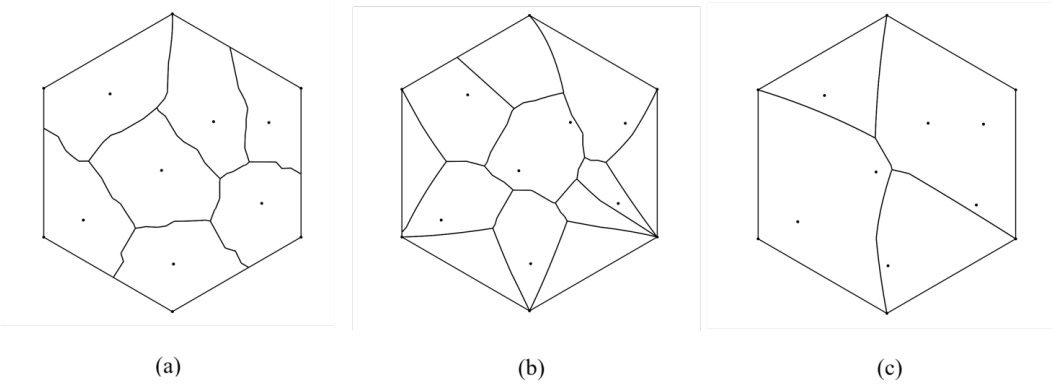


Figure 1 (a) First order Hilbert Voronoi (b) second order (c) final ($(n-1)^{th}$) order.

▶ **Lemma 1.** *The circumcenter of any three sites serves as a Voronoi vertex in both the k^{th} and $(k+1)^{th}$ order Hilbert Voronoi diagrams for some $0 < k < n-1$.*

Proof. Consider the circumcenter of three sites s_1, s_2, s_3 lying on the boundary of ball B, with $k-1$ sites of S in the ball. Perturbing B towards one of the sites will change which of s_1, s_2, s_3 is the k^{th} closest site to the center of B. Perturbing and growing the ball along one of the bisectors to include two sites will change which site is the $(k+1)^{st}$ closest. ◀

▶ **Lemma 2.** *Every bisector path between circumcenters belongs to exactly one k^{th} order cell.*

Proof. Consider the bisector between two sites $s_1, s_2 \in S$ lying on a k^{th}-order cell. A ball moving along the bisector and passing through both sites contains exactly $k-1$ other sites before reaching a circumcenter with a third site. Perturbing the ball off the bisector causes it to include either s_1 or s_2, shifting the center into the k^{th} order cell of one or the other. ◀

Our algorithm proceeds in three phases: Preprocessing computes all pairwise bisectors and all three site circumcenters parametrized along their respective bisectors. Looping over the bisectors lets us determine the diagram containment of the first bisector portion by checking the distance to each site. Traversing the bisector and updating diagram containment at each circumcenter yields a labeling of each bisector portion by the degree-order Hilbert Voronoi diagram it is part of, allowing us to display it quickly based on user input.

In the following we implicitly use three facts from [11,12]; first, we can order the boundary of Ω to calculate Hilbert distances in time $O(\log m)$, second we can order the boundary of Ω to calculate circumcenters in time $O(\log^3 m)$, and third, bisectors have complexity $O(m)$.

▶ **Corollary 3.** *Our algorithm takes time* $O(n^2 m \log m + n^3 \log^3 m + n^3 \log n \log m)$ *with space* $O(m \log m + n^3 + n^2 m)$.

Proof. Preprocessing takes $O(n^2 m \log m)$ time to compute all bisectors and $O(n^3 \log^3 m)$ time for computing all circumcenters (which we bucket by bisector). Sorting the circumcenters along the bisectors in the buckets takes $O(n^3 \log n)$.

We determine the containment of the first bisector component in $O(n \log n \log m)$ time per bisector. Then, we traverse the bisector, determining the containment of each vertex and each bisector segment in $O(n)$ time, since there are $O(n)$ circumcenters along each bisector. Since we do this for all n^2 bisectors, this takes $O(n^2(n \log n \log m + n))$. Our final runtime is $O(n^2 m \log m + n^3 \log^3 m + n^3 \log n \log m)$.

The storage consists of $O(m \log m)$ to order the boundary, $O(n^3)$ for circumcenters ordered along bisectors, and $O(n^2 m)$ for the bisectors, giving us $O(m \log m + n^3 + n^2 m)$ storage. ◀

▶ **Observation 4.** *In the Hilbert metric,* k^{th} *order Voronoi cells are not always star-shaped, a star-shaped region has an interior point from which all others are visible via a segment.*

Proof. This can be seen with the following Ω: a square with coordinates from (100, 100) to (300, 300) and sites at (160, 284.9), (140, 170), (130, 165), (180, 285). The second-order Voronoi generates a non-star-shaped region. This configuration is in general position. ◀

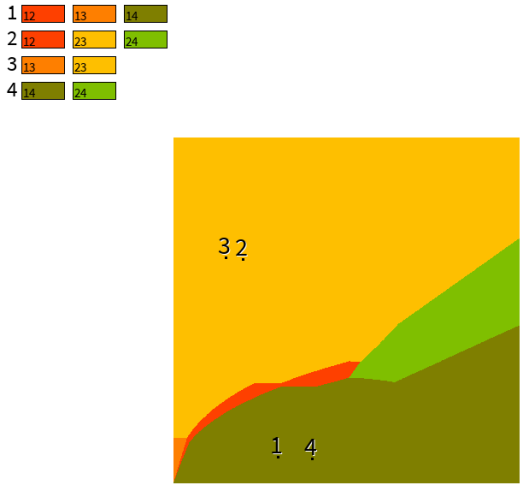


Figure 2 The red region, the Voronoi cell closest to 1 and 2, is not a star-shaped region.

We leverage our results on k^{th}-order cells to compute the Delaunay mosaic by computing a modified Fréchet mean (a point that minimizes the sum of distances) of the sites of each cell, and connecting them if their cells are adjacent.

3 Overlap and Outer Region

One notable property of the Hilbert metric is that three sites do not always have a circumcenter [9]. First we recall what it means to be a ball at infinity through two sites [11]. Then we define the overlap and outer regions which characterize circumcenter existence.

▶ **Definition 5** (Infinite Ball). *Given two sites $s_1, s_2 \in S$, consider balls passing through them with centers along the bisector parametrized by $t \in [0, 1]$. This yields two limit balls on the boundary of Ω: $B_0(p : q)$ at $t = 0$ and $B_1(p : q)$ at $t = 1$. (See Figure 3.)*

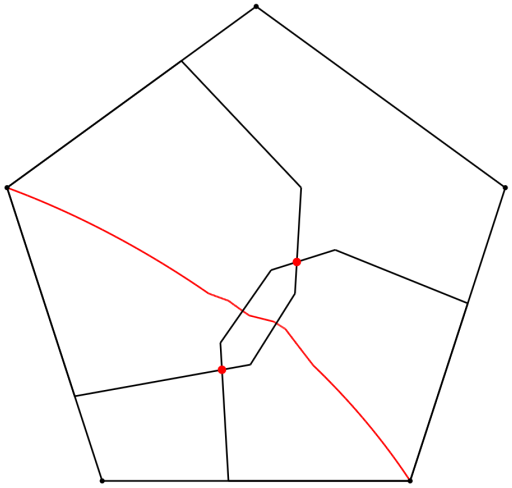


Figure 3 Two infinite balls along bisector through two sites (in red) generated by our software.

▶ **Definition 6** (Overlap Region). *Given two sites $s_1, s_2 \in S$, their overlap region, denoted $Z(p, q)$, is $B_0(p : q) \cap B_1(p : q)$.*

▶ **Definition 7** (Outer Region). *Given two sites $s_1, s_2 \in S$, their outer region, denoted $W(p, q)$, is $\operatorname{int}\Omega - (B_0(p : q) \cup B_1(p : q))$.*

The *overlap* is the region between two sites s_1, s_2 where there is no circumcenter between s_1, s_2, and p [11]. The *outer region* is the region that is not between the two sites where there is no circumcenter. In our software, we give users the option to display these regions dynamically. Users can generate these regions for any two sites, then move a third site between them to observe the behavior of the bisectors of the three sites.

4 Clustering

In 2019, Nielsen and Sun introduced clustering in the Hilbert simplex [17]. We expand on this by allowing users to interactively visualize clustering on arbitrary Hilbert polygonal geometries. We provide k-means and single linkage clustering (see Figure 4).

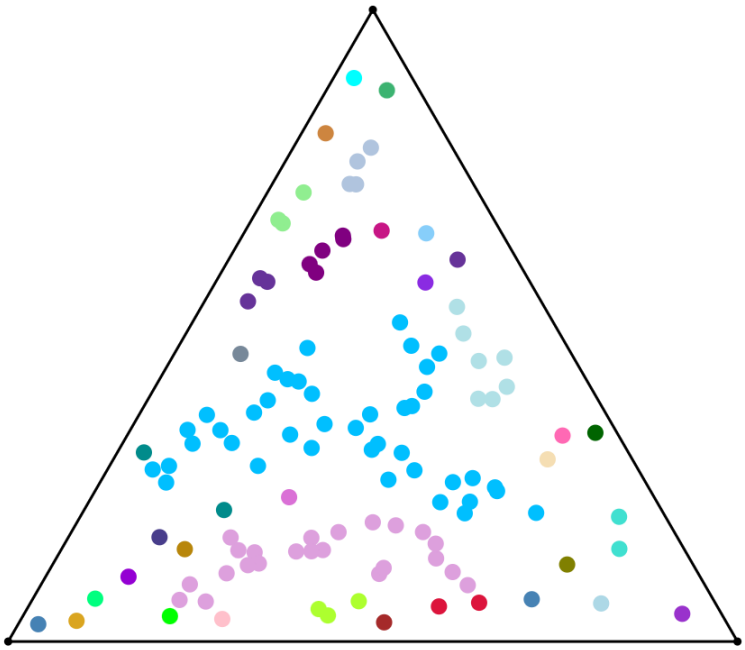


Figure 4 120 points in 38 clusters using single link clustering.

5 Conclusion

We have introduced the first software for visualizing higher-order Voronoi diagrams, Delaunay mosaics, clustering, overlap and outer regions in arbitrary convex Hilbert polygonal geometries.

References

1 Ahmed Abdelkader and David M. Mount. Economical Delone sets for approximating convex bodies. In *Proc. 16th Scand. Workshop Algorithm Theory*, pages 4:1–4:12, 2018. doi:10.4230/LIPIcs.SWAT.2018.4.

2 Ahmed Abdelkader and David M. Mount. Convex approximation and the Hilbert geometry. In *2024 Symposium on Simplicity in Algorithms (SOSA)*, pages 286–298. SIAM, 2024. doi:10.1137/1.9781611977936.26.

3 Hridhaan Banerjee, Soren Brown, June Cagan, Auguste H. Gezalyan, Megan Hunleth, Veena Kailad, Chaewoon Kyoung, Rowan Shigeno, Yasmine Tajeddin, Andrew Wagger, Kelin Zhu, and David M. Mount. Visualizing Higher Order Structures, Overlap Regions, and Clustering in the Hilbert Geometry. Software (visited on 2026-05-12). URL: https://andrew.wagger.net/hilbert/, doi:10.4230/artifacts.25998.

4 Hridhaan Banerjee, Soren Brown, June Cagan, Auguste H. Gezalyan, Megan Hunleth, Veena Kailad, Chaewoon Kyoung, Rowan Shigeno, Yasmine Tajeddin, Andrew Wagger, Kelin Zhu, and David M. Mount. Visualizing Higher Order Structures, Overlap Regions, and Clustering in the Hilbert Geometry. Audiovisual (visited on 2026-05-12). URL: https://youtu.be/yhJn--3Qgks, doi:10.4230/artifacts.25999.

5 Hridhaan Banerjee, Carmen Isabel Day, Auguste H. Gezalyan, Olga Golovatskaia, Megan Hunleth, Sarah Hwang, Nithin Parepally, Lucy Wang, and David M. Mount. Software For the Thompson and Funk Geometry. doi:10.4230/artifacts.23291.

6 Hridhaan Banerjee, Carmen Isabel Day, Megan Hunleth, Sarah Hwang, Auguste H Gezalyan, Olya Golovatskaia, Nithin Parepally, Lucy Wang, and David M Mount. On the heine-borel property and minimum enclosing balls. *arXiv preprint arXiv:2412.17138*, 2024. doi:10.48550/arXiv.2412.17138.

7 Cecilia Bohler, Panagiotis Cheilaris, Rolf Klein, Chih-Hung Liu, Evanthia Papadopoulou, and Maksym Zavershynskyi. On the complexity of higher order abstract voronoi diagrams. *Computational Geometry*, 48(8):539–551, 2015. doi:10.1016/j.comgeo.2015.04.008.

8 Cecilia Bohler, Rolf Klein, and Chih-Hung Liu. An efficient randomized algorithm for higher-order abstract voronoi diagrams. *Algorithmica*, 81(6):2317–2345, 2019. doi:10.1007/S00453-018-00536-7.

9 Madeline Bumpus, Caesar Dai, Auguste H. Gezalyan, Sam Munoz, Renita Santhoshkumar, Songyu Ye, and David M. Mount. Analysis of dynamic voronoi diagrams in the hilbert metric. *Proceedings of the 35th Canadian Conference on Computational Geometry (CCCG 2023) Montreal, Canada*, 2023.

10 Mercè Claverol, Andrea de las Heras Parrilla, Clemens Huemer, and Alejandra Martínez-Moraian. The edge labeling of higher order voronoi diagrams. *Journal of Global Optimization*, 90(2):515–549, 2024. doi:10.1007/S10898-024-01386-0.

11 Auguste H. Gezalyan, Soo H. Kim, Carlos Lopez, Daniel Skora, Zofia Stefankovic, and David M. Mount. Delaunay Triangulations in the Hilbert Metric. In Hans L. Bodlaender, editor, *19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024)*, volume 294 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SWAT.2024.25.

12 Auguste H. Gezalyan and David M. Mount. Voronoi diagrams in the Hilbert metric. In *39th International Symposium on Computational Geometry (SoCG 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.SoCG.2023.35.

13 Der-Tsai Lee. On k-nearest neighbor voronoi diagrams in the plane. *IEEE transactions on computers*, 100(6):478–487, 1982. doi:10.1109/TC.1982.1676031.

14 Bas Lemmens and Roger D. Nussbaum. Birkhoff's version of hilbert's metric and its applications in analysis, December 2014. doi:10.4171/147-1/10.

15 Mook Kwon Jung Minju Song and Hee-Kap Ah. Farthest-point voronoi diagrams in the hilbert metric. In *Proceedings of the 19th Algorithms and Data Structures Symposium (WADS 2025)*, 2025.

16 Frank Nielsen and Laetitia Shao. On balls in a Hilbert polygonal geometry (multimedia contribution). In *Proc. 33rd Internat. Sympos. Comput. Geom.*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 67:1–67:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.SoCG.2017.67.

17 Frank Nielsen and Ke Sun. Clustering in Hilbert's projective geometry: The case studies of the probability simplex and the elliptope of correlation matrices. In Frank Nielsen, editor, *Geometric Structures of Information*, pages 297–331. Springer Internat. Pub., 2019. doi:10.1007/978-3-030-02520-5_11.

18 Frank Nielsen and Ke Sun. Non-linear embeddings in Hilbert simplex geometry. In *Topological, Algebraic and Geometric Learning Workshops 2023*, pages 254–266. PMLR, 2023. URL: https://proceedings.mlr.press/v221/nielsen23a.html.

From Chaos to Continents: Voronoi-Based Procedural Terrain Generation with Hydrology and 3D Visualization

Batsambuu Batbold
Department of Mathematics, Statistics, and Computer Science, Macalester College, St. Paul, MN, USA

Lori Ziegelmeier
Department of Mathematics, Statistics, and Computer Science, Macalester College, St. Paul, MN, USA

Abstract

Procedural content generation often employs grid-based methods to create virtual environments. We present a pipeline that utilizes Voronoi diagrams and Lloyd's Relaxation to construct an irregular mesh for terrain generation. We implement a customizable "Land Anchor" system combined with Perlin noise to determine landmass shapes, distinct from standard radial distribution methods. Furthermore, we simulate hydrology using priority-flood routing on the Voronoi edges and assign biomes via a Gaussian-smoothed Whittaker classification. The full pipeline is exposed through an interactive application that enables real-time parameter tuning and terrain export, and resulting geometric data is extruded in Blender to produce a 3D terrain model.

2012 ACM Subject Classification Computing methodologies → Image-based rendering; Theory of computation → Computational geometry

Keywords and phrases Procedural Content Generation, Voronoi Diagrams, Lloyd's Relaxation, Perlin Noise, Blender

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.101

Category Media Exposition

Related Version *Blog Post*: `https://basabu1.github.io/Map-Generation/` [2]

Supplementary Material
Software: `https://github.com/BaSaBu1/Map-Generation` [4]
archived at `swh:1:dir:0ce321b8c3e39d9e7d85f1d6855e3c43cf34a476`
InteractiveResource (Web Application): `https://basabu1-map-generation-app-fut9qy.streamlit.app/` [3]

Funding *Lori Ziegelmeier*: Supported by the NSF award BCS-2318171.

1 Introduction

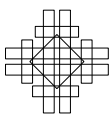
Procedural Content Generation (PCG) creates virtual environments algorithmically [15, 10]. While grid-based approaches are computationally efficient, they often produce rectilinear artifacts. Polygonal map generation based on Voronoi diagrams offers an alternative for constructing irregular meshes; our work is inspired by the framework detailed in [13].

Existing polygonal implementations to generate terrains, such as [13], commonly employ a radial island mask biasing land generation toward the map center and utilize simplified hydrology models. In this work, we extend this polygonal approach to support flexible landmass design as well as 3D visualization. Our approach introduces three primary modifications:

1. **Geometry Generation:** Instead of a radial mask, we implement a customizable "Land Anchor" system combined with Perlin noise, allowing for the generation of non-convex landmasses and archipelagos.


© Batsambuu Batbold and Lori Ziegelmeier;
licensed under Creative Commons License CC-BY 4.0





2. **Hydrology Simulation:** We use a priority-flood algorithm to simulate flow accumulation for river generation along the Voronoi edges.
3. **3D Visualization:** We extend the 2D graph data into a 3D pipeline, converting the mesh into heightmaps for rendering in Blender.

We discuss each step of our approach in the following sections.

2 Geometric Foundation

2.1 Voronoi Diagrams and Duality

To create an irregular mesh, we partition the plane using a *Voronoi diagram* [8]. Given a set of points S as sites, the Voronoi region V_p corresponding to a site $p \in S$ contains all points in the plane closer to p than to any other site in S.

We construct this mesh via its dual graph, the *Delaunay triangulation*. We utilize the `delaunator` library [12] to efficiently compute the triangulation. The circumcenters of the resulting Delaunay triangles serve as the vertices of the Voronoi polygons, while the edges between them define the adjacency relationships between cells.

2.2 Lloyd's Relaxation

Voronoi sites are generated uniformly at random, though naive sampling produces uneven distributions with high variance in cell area and edge length. To regularize the mesh, we apply *Lloyd's Relaxation* [11], which iteratively converges to a Centroidal Voronoi Tessellation [9]. This redistributes sites more evenly, producing cells that are relatively uniform in area while retaining irregular boundaries; see Figure 1.

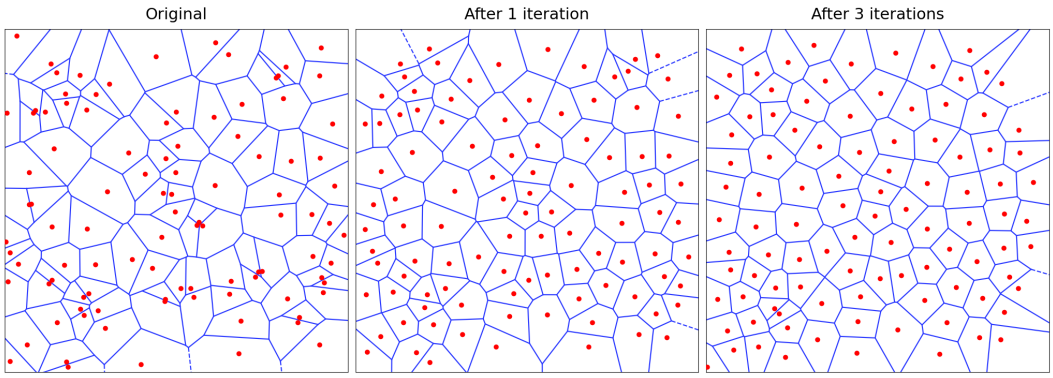


Figure 1 (Left) Voronoi mesh generated from random points. (Right) Uniform mesh resulting from 3 iterations of Lloyd's Relaxation.

3 Procedural Shaping

3.1 Elevation: Noise and Anchors

To generate elevation in our terrain, we use *Perlin noise* [14] implemented in [5], which provides spatially coherent values unlike uncorrelated sampling. We introduce a *Land Anchor* system to control terrain shape. Rather than a single radial mask for island formation, our method generalizes to N arbitrary anchor points. The final elevation E of a cell with Voronoi

site at (x, y) is defined as $E_{final} = \text{Perlin}(x, y) - (d_{min}/R)^2$ where d_{min} is the Euclidean distance to the nearest land anchor, and R is the map radius. This approach concentrates high elevation values around specific focal points while preserving the irregular, jagged nature of the noise along the coastlines; see Figure 2. Subsequently, a piecewise power curve is applied to the elevation data to compress low-lying areas into flat beaches and exponentially scale higher values into steep mountain peaks.

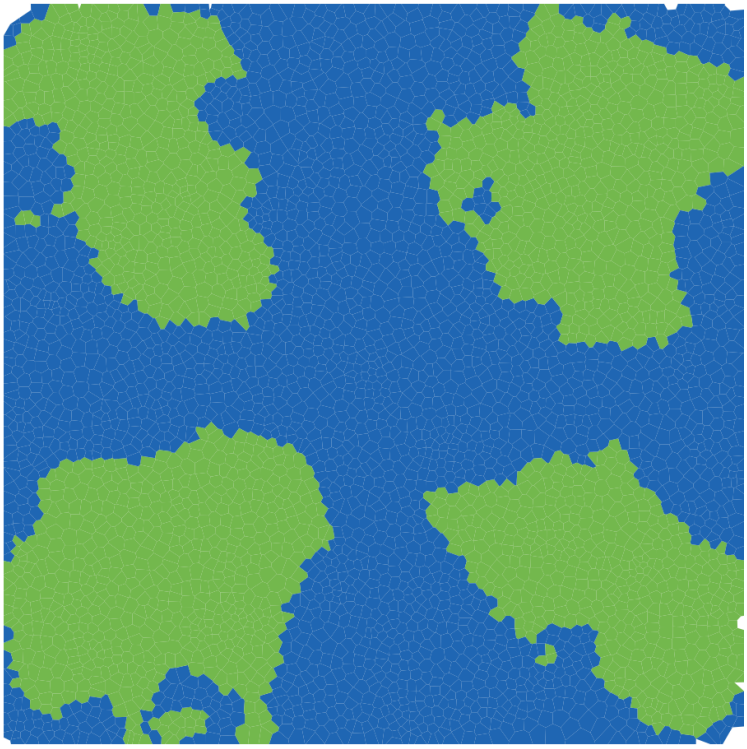


Figure 2 Non-radial archipelago layout generated using a four-land anchor configuration.

3.2 Biomes and Gaussian Smoothing

To texture the terrain, we simulate moisture using a secondary Perlin noise map modified by proximity to water. Cells are classified into biomes (e.g., Desert, Forest, Snow) using a *Whittaker Biome Diagram* [17], which maps $(Elevation, Moisture)$ pairs to terrain types.

Discrete biome classification can result in abrupt transitions between distinct zones; see Figure 3 (left). To mitigate this, we implement a smoothing pipeline:

1. Generate a 256×256 lookup table representing the Whittaker classification.
2. Apply SciPy's [16] *Gaussian Blur* ($\sigma = 8$) to the table to generate boundary gradients.
3. Assign cell colors using bilinear interpolation on this blurred table.

This results in gradient transition zones, such as a grassland buffer separating a forest from a beach; see Figure 3 (right).

3.3 Hydrology: Priority-Flood Routing

We simulate hydrology by utilizing the graph structure of the mesh, defining river channels along Voronoi edges. A *priority-flood algorithm* [1] is implemented to model flow dynamics:

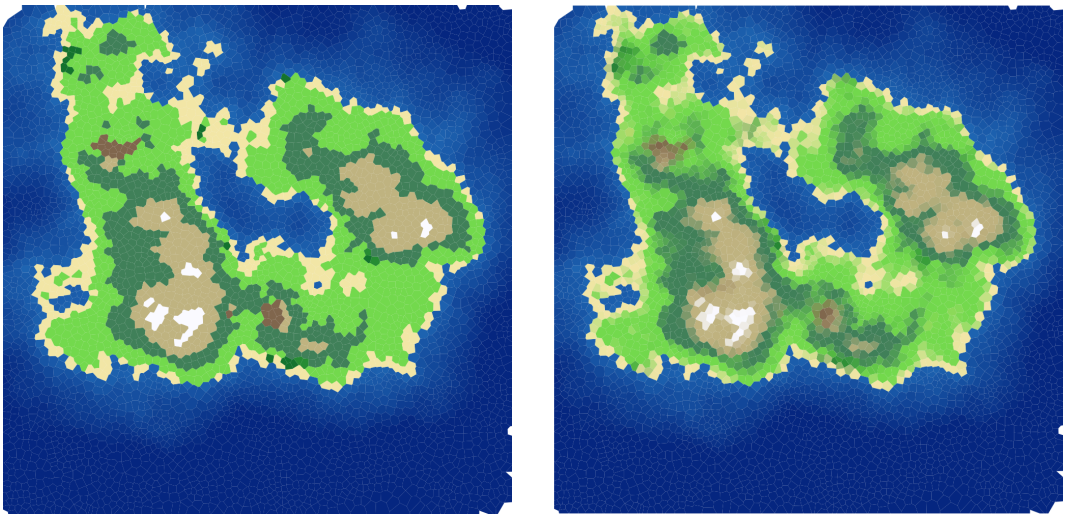


Figure 3 Biome distribution using rigid classification (left) and Gaussian-smoothed gradients between distinct zones (right).

1. **Flow Routing:** We construct a directed graph where each vertex connects to its lowest neighbor, creating a continuous path to the ocean.
2. **Flow Accumulation:** We traverse the graph from local maxima to the coast, aggregating flow values at each step.

The resulting accumulation values determine the width of the river segments, generating a river network structure; see Figure 4.

4 3D Visualization and Results

To visualize the generated terrain in three dimensions, we export the graph data using the Python Imaging Library [7]. We generate three primary texture maps:

- **Heightmap:** A 16-bit grayscale PNG representing the elevation data.
- **Colormap:** An RGB image representing the Gaussian-smoothed biome colors.
- **River Mask:** A binary mask isolating river pixels for shader application.

These textures are imported into Blender [6] to construct the 3D mesh. We apply a displacement modifier to a high-resolution plane to deform the geometry according to the heightmap. A custom shader utilizes the river mask to assign distinct material properties (specularity and roughness) to water surfaces. Figure 5 displays an example final output.

5 Conclusion

We present a pipeline for procedural terrain generation (see project webpage [2], interactive application [3] with example in Figure 6, and github repo [4]) that constructs a Voronoi mesh regularized via Lloyd's relaxation. Land shape was determined through a Land Anchor system with Perlin noise, and hydrology was simulated using a graph-based priority-flood algorithm. The mesh was exported to Blender for 3D rendering. The system produced plausible terrain, though the current Python implementation exhibited performance limitations at high polygon counts ($N > 50{,}000$). Future work includes computational optimization and atmospheric extensions (e.g., wind and temperature modeling) to support richer biome variation.

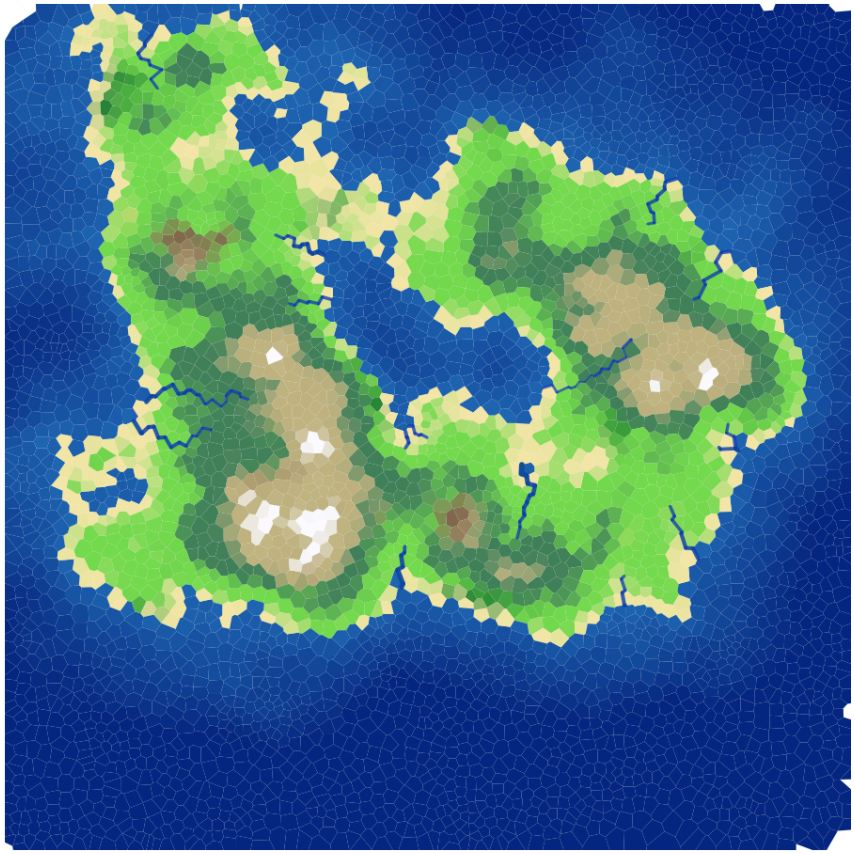


Figure 4 Hydrology simulation with river networks flowing along Voronoi edges.

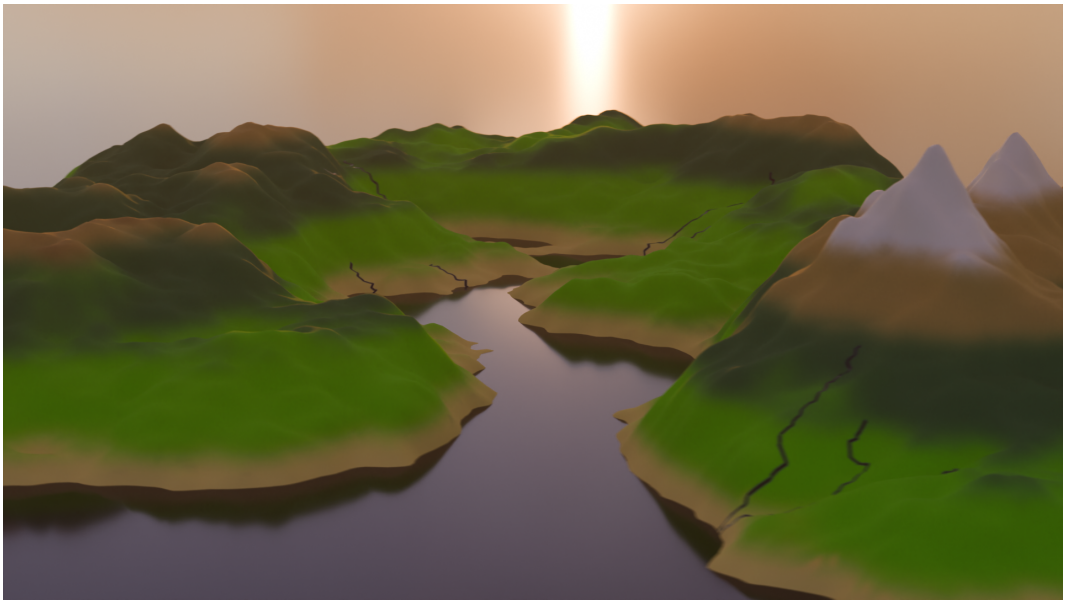


Figure 5 Example 3D terrain rendered in Blender.

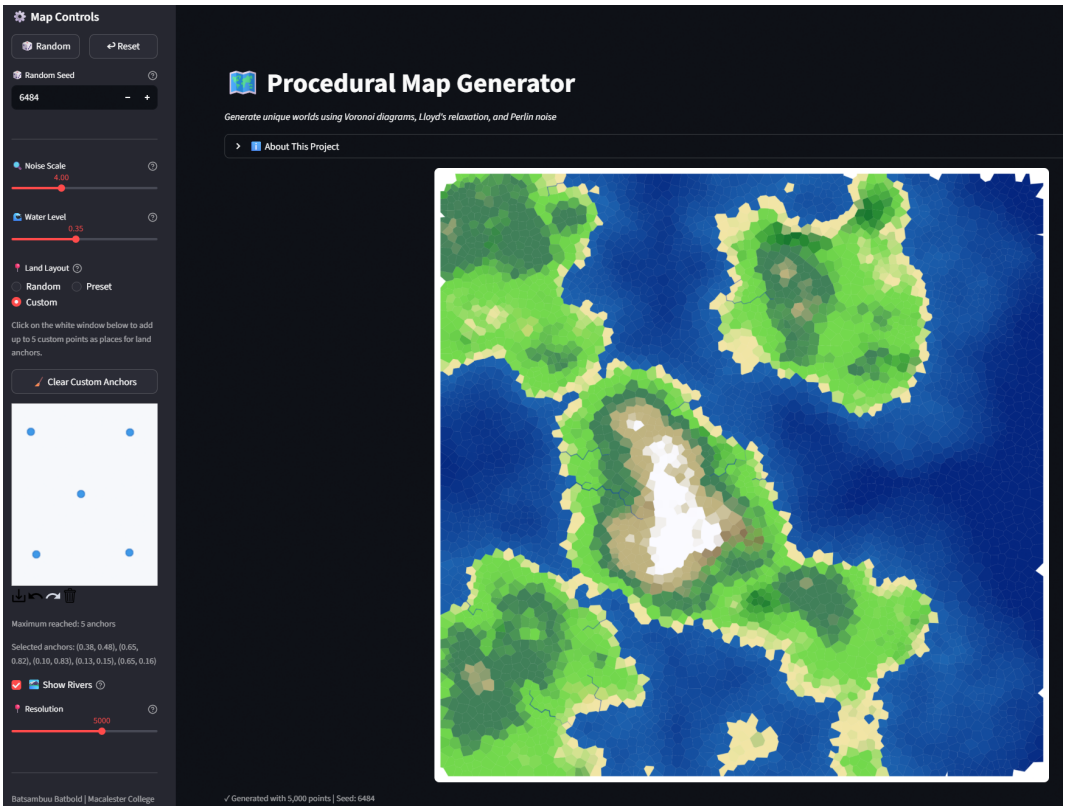


Figure 6 Screenshot of the Streamlit application [3], with sliders to specify parameters.

References

1 Richard Barnes, Clarence Lehman, and David Mulla. Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *Computers & Geosciences*, 62:117–127, 2014. doi:10.1016/j.cageo.2013.04.024.

2 Batsambuu Batbold. From chaos to continents: Procedural terrain generation. https://basabu1.github.io/Map-Generation/, 2025. Accessed: 2026-02-17.

3 Batsambuu Batbold. Map generation interactive app. https://basabu1-map-generation-app-fut9qy.streamlit.app/, 2025. Accessed: 2026-02-21.

4 Batsambuu Batbold. Map generation source code. https://github.com/BaSaBu1/Map-Generation, 2025. Accessed: 2026-02-21.

5 Alexandre Beyeler. Pnoise: A Pure Python Perlin Noise Generator. https://pypi.org/project/pnoise/, 2022. PyPI package.

6 Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2025. URL: http://www.blender.org.

7 Alex Clark. Pillow (pil fork) documentation, 2024. Python Imaging Library. URL: https://pillow.readthedocs.io/.

8 Satyan L Devadoss and Joseph O'Rourke. *Discrete and computational geometry*. Princeton university press, 2025.

9 Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999. doi:10.1137/S0036144599352836.

10 Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1):1–22, 2013. doi:10.1145/2422956.2422957.

11 S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. doi:10.1109/TIT.1982.1056489.

12 Mapbox. Delaunator: A fast library for delaunay triangulation of 2d points, 2024. Accessed: 2026-02-17. URL: https://github.com/mapbox/delaunator.

13 Amit Patel. Polygonal map generation for games, 2010. Red Blob Games. Accessed: 2026-02-17. URL: http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/.

14 Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985. doi:10.1145/325334.325247.

15 Noor Shaker, Julian Togelius, and Mark J Nelson. *Procedural content generation in games*. Springer, 2016. doi:10.1007/978-3-319-42716-4.

16 Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, A. H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020.

17 Wyoming Biodiversity Information System. Whittaker biome diagram guide. https://gveg.wyobiodiversity.org/application/files/7916/4641/2117/Whittaker_Diagram_Guide.pdf, 2020. Accessed: 2026-02-16.

Tracking a Set of Moving Objects with Minimal Peak Power

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
L3S Research Center, Hannover, Germany

Malte Hoffmann
Department of Computer Science, TU Braunschweig, Germany

Chek-Manh Loi
Department of Computer Science, TU Braunschweig, Germany

Michael Perk
Department of Computer Science, TU Braunschweig, Germany

Abstract

A common sensing problem is to use a set of stationary tracking locations to monitor a collection of moving devices. Given n objects that need to be tracked, each following its own trajectory, and m stationary traffic control stations, each with a sensing region that can be changed over time; how should we adjust the individual sensor ranges in order to optimize energy consumption? We illustrate how to combine geometric insights with mathematical optimization to find optimal solutions for the min max variant of the problem, which aims at minimizing peak power consumption. Instances with 500 moving objects and 25 stations can be solved in the order of seconds for scenarios that take minutes to play out in the real world, demonstrating real-time capability of our methods.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Set cover, kinetic problems, geometric optimization, exact optimization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.102

Category Media Exposition

Related Version *Full Version*: https://arxiv.org/abs/2603.05286 [14, 15]
Video Link: https://youtu.be/YoLfsEbJ04Q

Supplementary Material *Software (and Data)*: https://doi.org/10.5281/zenodo.18955730 [9]

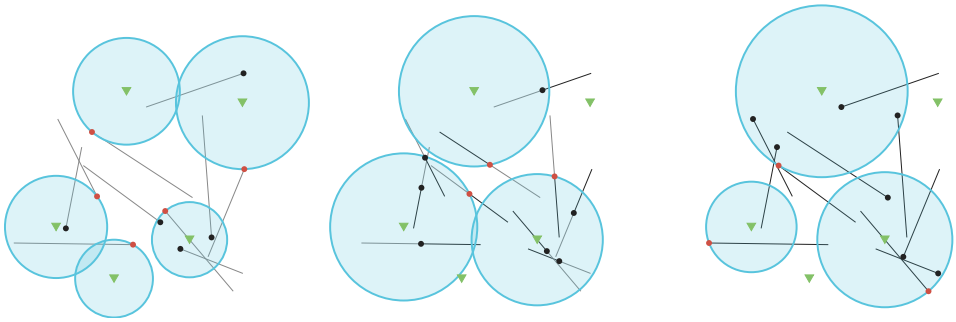
Funding This work was supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) as part of project Computational Geometry: Solving Hard Optimization Problems (CG:SHOP) – 444569951.

1 Introduction

Keeping track of a collection of moving objects is a fundamental problem with a long history in Computational Geometry. In air traffic control, stationary centers monitor planes with powerful tracking devices to coordinate overall motion. With the growing ubiquity of drones, air traffic control needs further development [4, 16], such as the use of less powerful tracking stations: How should we adjust sensing radii over time to minimize power consumption?

We illustrate exact and heuristic methods for the min max variant of the Kinetic Disk Cover problem (KDC): Given a set of m stationary sensors at positions $\mathcal{Y} \subset \mathbb{R}^2$, as well as a set $\mathcal{P}$ of n moving objects, each moving along a linear trajectory $p_i(t)$ over the time interval $t \in [0, 1]$. How should we adjust the sensor range $r_i(t)$ to each center $y_i \in \mathcal{Y}$ at each time t, such that all objects are always within range of some sensor, and the maximum sum of areas $\max_{t\in[0,1]} \sum_{i=1}^{m} \pi r_i(t)^2$ of all disks is minimized?

© Sándor P. Fekete, Malte Hoffmann, Chek-Manh Loi, and Michael Perk;
licensed under Creative Commons License CC-BY 4.0



(a) $t = 0$ **(b)** $t = 0.5$ **(c)** $t = 1$

Figure 1 A solution for an instance of the Kinetic Disk Covering Problem for $n = 10$ and $m = 5$. Stations are shown as green triangles, points move along the line segments.

Related work. The stationary Disk Cover problem (DC) [1] and its generalization, the non-uniform minimum-cost multi-cover (MCMC) problem [12, 13] are known to be NP-hard. Recently, there has been some work on optimally multi-covering fixed points with disks of varying sizes [11]. Other work examines multi-covering points with the minimum number of unit disks, with centers at arbitrary locations [10]. Research on fundamental concepts for kinetic data structures [2] enabled algorithms for k-center and k-means problem for both kinetic stations and objects [3].

2 Algorithmic approach

We solve the kinetic problem with a primal–dual, iterative strategy. By combining exact stationary solutions with good heuristics for obtaining overall kinetic solutions, we generate a sequence of tighter and tighter upper and lower bounds and therefore provably near-optimal or even optimal solutions.

Our approach starts from a feasible, stationary solution at time $t = 0$. We can extend any solution to the DC problem at time t to a solution for the KDC problem over time by maintaining the same assignment of objects to stations and altering the supporting point of each station as needed to ensure coverage. This produces a feasible solution for all $t \in [0, 1]$, yielding an upper bound for the optimal objective value. The upper bound can be further improved by considering the times at which it is advantageous to switch objects between covering stations, so-called *handover points*. In the full version of our paper [14], we establish the following; this can be generalized to three or more stations.

▶ **Lemma 1.** *In $\mathcal{O}(m^2n^3)$ time, all $\mathcal{O}(m^2n^3)$ possible handovers of objects between two stations (when extending a stationary solution to $t \in [0, 1]$) can be computed.*

On the other hand, we can consider the time t' at which the peak total area is attained; an optimal solution for the static DC problem at t', which can be computed with the help of an Integer Programming (IP) solver, provides a lower bound for the kinetic problem.

$$\begin{aligned} \text{minimize} \quad & \pi \cdot \sum_{d_i \in C} r_i^2 x_i \\ \text{subject to} \quad & \sum_{\substack{d_i \in C \\ p_j \in d_i}} x_i \geq 1, \qquad \forall p_j \in \mathcal{P} \\ & x_i \in \{0, 1\}, \quad \forall d_i \in C \end{aligned}$$

In this formulation, C is the set of all possible disks that can be formed by a station and an object, r_i is the radius of the disk d_i, and x_i indicates whether d_i is selected. A solution for the stationary problem can again be extended to an overall solution for the kinetic problem. Combining it with the previous kinetic solution by extracting the lower envelope of both solutions yields an improved feasible solution, and thus a better upper bound, see Figure 2.

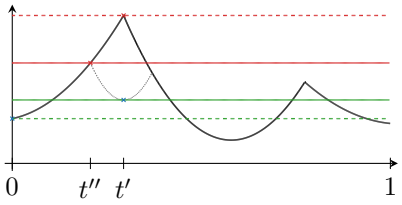


Figure 2 Improvement of the static solution at time t'. Green and red lines denote lower and upper bounds on the optimal solution.

3 Evaluation

We carried out extensive experiments for solving a wide range of benchmark instances; see our full paper [14, 15] for a detailed description.

We considered the following benchmark sets.

`fix_n` Fix number of objects to $n = 500$. For $m \in \{5, 10, \ldots, 50\}$ we generate 10 instances.

`fix_m` Fix number of sensors to $m = 25$. For $n \in \{50, 100, \ldots, 500\}$ we generate 10 instances.

`pub` Instances from well-known publicly available benchmarks used in [8], such as point sets from the CG:SHOP challenges [5, 6] and [7, 18, 19]. We scaled point sets to a fixed size and sampled $m = 25$ centers and point pairs (for trajectories) at random.

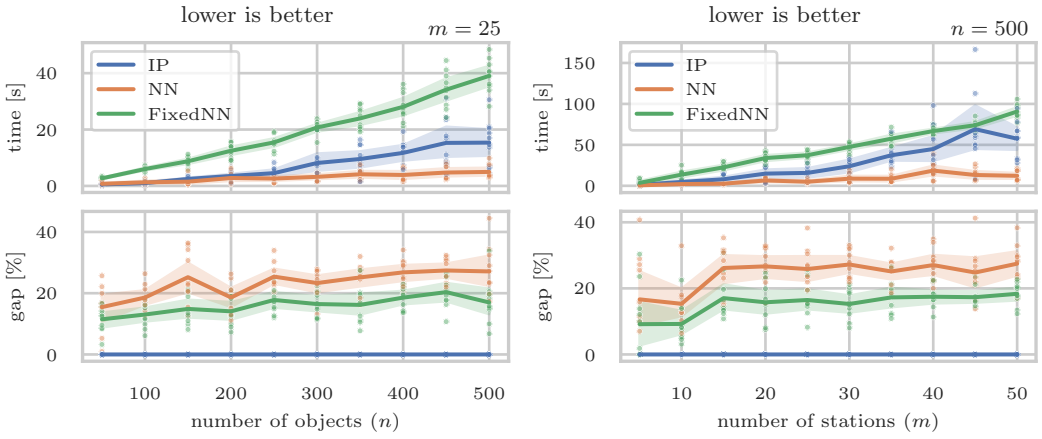


Figure 3 Performance of `IP`, `NN` and `FixedNN` on instances from the `fix_m` and `fix_n` datasets. While `IP` is significantly slower than `NN`, it produces optimal solutions for all instances. `FixedNN` is slightly better in terms of solution quality than `NN` but significantly slower.

Influence of m and n. We evaluate the influence of the number of objects n and the number of stations m on the performance of the algorithm. We execute the algorithm from Section 2 with both an IP solver and a nearest neighbor heuristic to solve the DC problem

in each iteration. We denote the IP-based algorithm as `IP` and the nearest neighbor based algorithm as `NN`. Additionally, we implemented another heuristic that divides the time $[0, 1]$ into $k = 10$ evenly spaced intervals and computes the nearest neighbor solution at each border. The algorithm `FixedNN` then extends all solutions and reports the lower envelope. Figure 3 shows the performance of the different algorithms. Overall, `IP` is able to solve all instances to optimality in a matter of seconds, demonstrating real-time capability for scenarios that take minutes to play out in the real world.

Instances from Literature. We further assess the algorithm's performance on the `pub` instance set, as shown in Figure 4 and illustrated by Figure 5. The presence of degeneracies (such as equidistant points and collinearities), originating from real-world data, led to various numerical challenges in our implementation. To address these issues, we developed an alternative version of `IP` utilizing exact number types for computations; see the full paper [14, 15] for details. Our results show that 290 out of 302 instances with up to 700 objects could be solved to provable optimality within 600 s of computation time.

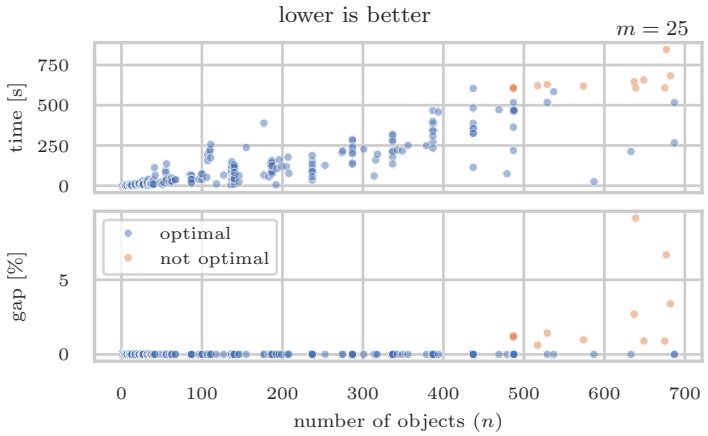
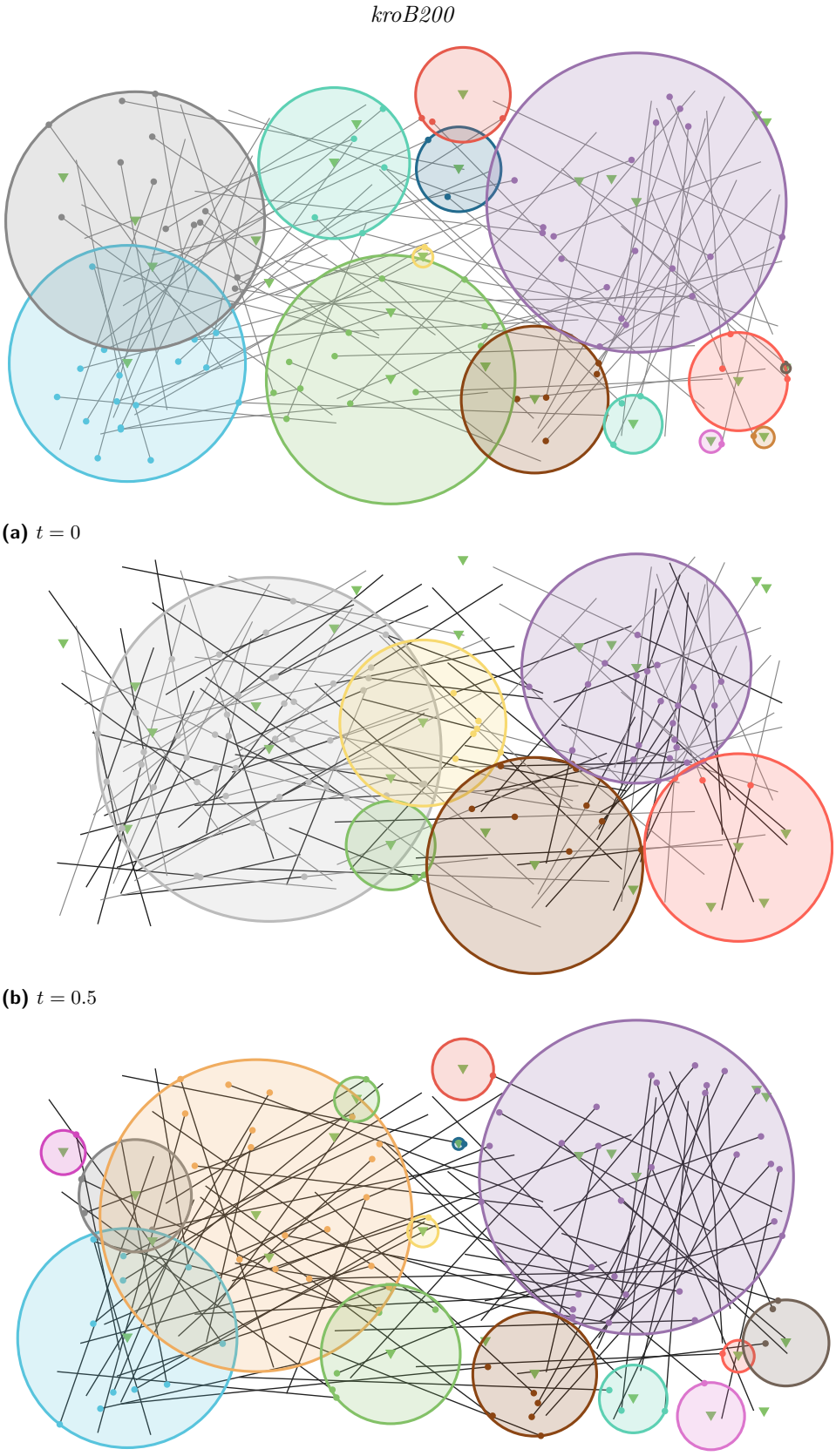


Figure 4 Performance of the exact version of `IP` on the `pub` dataset. Despite the variety and degeneracies, almost all instances can be solved to provable optimality.

4 The video

We start by introducing and motivating the KDC problem before describing the primal-dual approach, illustrating the computation of upper and lower bounds as well as the overall iterative method. We proceed by showing some of the involved technical components and evaluations, concluding with a benchmark instance with 87 objects and 25 stations derived from a point set of the well-known TSPLIB benchmark library, as shown in Figure 5.

Overall video production was based on an integrated tool chain: Visual elements were produced with the free, open-source software Manim [20] for creating mathematical animations, which also supports conversion from scripted text to synchronized voiceover and animation. Voiceover was generated with the help of Microsoft Azure Speech [17].



(a) $t = 0$

(b) $t = 0.5$

(c) $t = 1$

Figure 5 A solution of an instance from `pub` instance set based on *kroB200* TSPLIB [18].

References

1 Helmut Alt, Esther M. Arkin, Hervé Brönnimann, Jeff Erickson, Sándor P. Fekete, Christian Knauer, Jonathan Lenchner, Joseph S. B. Mitchell, and Kim Whittlesey. Minimum-cost coverage of point sets by disks. In *Symposium on Computational Geometry (SoCG)*, pages 449–458, 2006. doi:10.1145/1137856.1137922.

2 Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. *J. Algorithms*, 31(1):1–28, 1999. doi:10.1006/JAGM.1998.0988.

3 Sergei Bespamyatnikh, Binay K. Bhattacharya, David G. Kirkpatrick, and Michael Segal. Mobile facility location. In *International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 46–53, 2000. doi:10.1145/345848.345858.

4 Adam P. Cohen, Susan A. Shaheen, and Emily M. Farrar. Urban air mobility: History, ecosystem, market potential, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):6074–6087, 2021. doi:10.1109/TITS.2021.3082767.

5 Erik D Demaine, Sándor P Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing convex partitions for point sets in the plane: The CG:SHOP Challenge 2020. *arXiv preprint arXiv:2004.04207*, 2020. arXiv:2004.04207.

6 Erik D Demaine, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Area-optimal simple polygonalizations: The CG Challenge 2019. *Journal of Experimental Algorithmics (JEA)*, 27(2):1–12, 2022. doi:10.1145/3504000.

7 Günther Eder, Martin Held, Steinþór Jasonarson, Philipp Mayer, and Peter Palfrader. Salzburg database of polygonal data: Polygons and their generators. *Data in Brief*, 31:105984, 2020. doi:10.1016/j.dib.2020.105984.

8 Sándor P. Fekete, Phillip Keldenich, and Michael Perk. Exact Algorithms for Minimum Dilation Triangulation. In *Symposium on Computational Geometry (SoCG)*, pages 48:1–48:18, 2025. doi:10.4230/LIPIcs.SoCG.2025.48.

9 Sándor Fekete, Malte Hoffmann, Chek-Manh Loi, and Michael Perk. Drone air traffic control: Tracking a set of moving objects with minimal power, 2026. doi:10.5281/zenodo.18955730.

10 Xuening Gao, Longkun Guo, and Kewen Liao. Fast approximation algorithms for multiple coverage with unit disks. In *Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 185–193, 2022. doi:10.1109/WoWMoM54355.2022.00037.

11 Mariem Guitouni, Chek-Manh Loi, Sándor P. Fekete, Michael Perk, and Aaron T. Becker. Multi-covering a point set by m disks with minimum total area. In *ICRA*, pages 3000–3006. IEEE, 2025. doi:10.1109/ICRA55743.2025.11127835.

12 Ziyun Huang, Qilong Feng, Jianxin Wang, and Jinhui Xu. PTAS for minimum cost multi-covering with disks. In *Symposium on Discrete Algorithms (SODA)*, pages 840–859, 2021. doi:10.1137/1.9781611976465.53.

13 Ziyun Huang, Qilong Feng, Jianxin Wang, and Jinhui Xu. PTAS for minimum cost multicovering with disks. *SIAM Journal on Computing*, 53(4):1181–1215, 2024. doi:10.1137/22M1523352.

14 Chek-Manh Loi, Michael Perk, Malte Hoffmann, and Sándor P. Fekete. Drone air traffic control: Tracking a set of moving objects with minimal power. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026. To appear.

15 Chek-Manh Loi, Michael Perk, Malte Hoffmann, and Sándor Fekete. Drone air traffic control: Tracking a set of moving objects with minimal power, 2026. doi:10.48550/arXiv.2603.05286.

16 Franco Mazzenga, Romeo Giuliano, and Alessandro Vizzarri. 5G-based synchronous network for air traffic monitoring in urban air mobility. *IEEE Access*, 12:188542–188559, 2024. doi:10.1109/ACCESS.2024.3513212.

17 Microsoft. Azure Speech in Foundry Tools, 2026. URL: https://azure.microsoft.com/en-us/products/ai-foundry/tools/speech.

18 G. Reinelt. TSPLIB–A Traveling Salesman Problem Library. *ORSA Journal of Computing*, 3(4):376–384, 1991. doi:10.1287/ijoc.3.4.376.

19 Andre Rohe. VLSI data set. URL: https://www.math.uwaterloo.ca/tsp/vlsi/index.html.

20 The Manim Community Developers. Manim – Mathematical Animation Framework, 2026. URL: https://www.manim.community/.

Scalable Algorithmic Methods for Simulating Heavy-Rain Events

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany
L3S Research Center, Hannover, Germany

Phillip Keldenich
Department of Computer Science, TU Braunschweig, Germany

Michael Perk
Department of Computer Science, TU Braunschweig, Germany

Tobias Wallner
Department of Computer Science, TU Braunschweig, Germany

Abstract

We motivate and demonstrate simulation and evaluation of large-scale, fine-grained hydrodynamic flows, triggered by heavy-rain events. We show significant progress for simulating time-dependent, high-resolution runoff in large-scale heavy-rain scenarios, based on different geometry-based algorithmic speedup techniques. This enables us to address a second challenge: How can we deal with the instability of precipitation events, which are notoriously difficult to predict with good accuracy?

2012 ACM Subject Classification Theory of computation → Computational geometry; General and reference → Experimentation

Keywords and phrases Heavy-rain events, hydrodynamic flows, scalable simulation, applied computational geometry, sensitivity analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.103

Category Media Exposition

Funding The work described here was part of project EXDIMUM, supported by the German Federal Minstery of Research, Technology and Space, grant number 02WEE1631A.

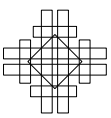
Acknowledgements We thank our cooperation partners at SCALGO, in particular, Morten Revsbæk, Thomas Mølhave and Matthias Rav. In memoriam Lars Arge.

1 Introduction

Throughout human history, natural disasters have been a significant threat [17]. Both frequency and extent of heavy-rain events have increased in recent time, as observed by [10] in 2020 for Germany, Thus, the development of efficient and stable tools for hydrodynamic modeling has been an active field for decades, both in research and in product development [12]. The use of high-resolution elevation models is important for capturing terrain details that can have a major impact on the runoff behavior of surface water [11]; however, the high-resolution simulation is computationally highly intensive, making it difficult to employ it for larger areas, especially in early warning systems [8, 13]. Accordingly, many existing tools offer different compromises between modeling details, preprocessing time, and computation time. Scalable approaches for these important tasks are based on combining a number of algorithmic tools, including methods from computational geometry. While previous success stories (most notably by Arge et al. [1–7, 14]) have made significant progress on stationary flood levels in largely flat terrain, the problem of flooding with time-dependent water flow in sloped terrain remains computationally demanding even with modern tools and commercial software.



© Sándor P. Fekete, Phillip Keldenich, Michael Perk, and Tobias Wallner;
licensed under Creative Commons License CC-BY 4.0



Beyond all these efforts for scalable simulation methods, a second, potentially even bigger challenge looms: How can we deal with the unreliable nature of one of the most crucial inputs for any such model – the precipitation data, in time and space? Acquiring this time-dependent input with an accuracy similar to that of a finely resolved, static terrain is highly challenging, and even more difficult when it comes to forecasting such precipitation. As a consequence, computational methods from meteorology, computational physics or modern AI will likely continue to struggle to obtain reliable, fine-grained input, due to the instability of critical weather phenomena. This also implies that real-time intervention, as well as preemptive measures, will continue to suffer from a lack of reliable forecasting of the ensuing runoff.

2 Study Area and Hydraulic Modeling

While our methodological work is applicable to any region with sufficiently detailed terrain data, the main focus area of our work is a 370 km^2 polygonal domain centered on Goslar, Germany, and lies entirely within the federal state of Lower Saxony (Niedersachsen). Elevations increase from ca. 100 m above sea level (a.s.l.) in the Oker valley north of Goslar to almost 800 m a.s.l. in the southern sector according to Copernicus EU-DEM v1.1 [9].

Testing, improving and benchmarking of our simulations was performed with the commercial solver TUFLOW HPC version 2025.1.0, which is a 1D/2D hydraulic modeling software that solves the depth-averaged Navier-Stokes equations, also known as the shallow water equations, to simulate the flow of water over a terrain. TUFLOW uses an explicit finite volume solution with a square grid, whose cells might vary in size. Further details on the numerical scheme can be found in the TUFLOW manual [16] and in our full paper [15].

3 Enhancing Simulations

In general, there are two main challenges when simulating large areas with high resolution. Firstly, the memory footprint of the simulation becomes prohibitively large, making it infeasible to run the simulation on current hardware. Secondly, the runtime of the simulation also eventually grows beyond a reasonable time frame. This is often countered by using coarser resolution models, which can lead to a loss of accuracy in crucial areas.

Our work [15] is based on two algorithmic improvements. Firstly, we have developed a subdivision algorithm that partitions the terrain into smaller regions that can be simulated independently, thus reducing the memory footprint of the simulation; see Figure 1. Secondly, we use a quadtree refinement approach that (given areas and desired resolution as inputs) automatically refines the mesh in critical regions. A key contribution of our work is the analysis of *what* regions require refinement and *how much* refinement is necessary to maintain accuracy while minimizing computational cost; see Figure 2. Both approaches can be combined to enable large-scale simulations on modern hardware.

4 Outcomes

There are two main outcomes of our work: (I) progress for simulating time-dependent, high-resolution runoff in large-scale heavy-rain scenarios, based on the two algorithmic approaches described in Section 3. In combination with state-of-the-art commercial software and powerful computing devices, these approaches can achieve simulation times that are significantly shorter than the duration of the simulated events for modestly sized regions (on the order of tens of square kilometers) at meter-scale resolution. The exact times depend on

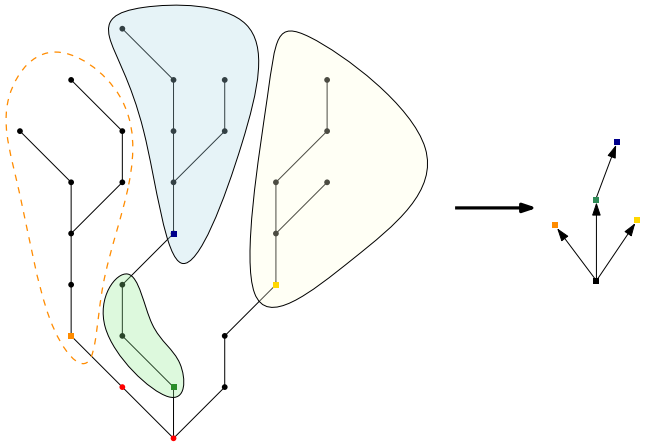


Figure 1 (Left) Subdividing a large hydrodynamic flow graph into smaller, dependency-aware polygonal regions bounded by an inflow threshold, enabling memory-efficient and parallelizable simulation. (Right) Dependency graph of the flow components after subdivision. Each node represents a component and edges indicate dependencies between them.

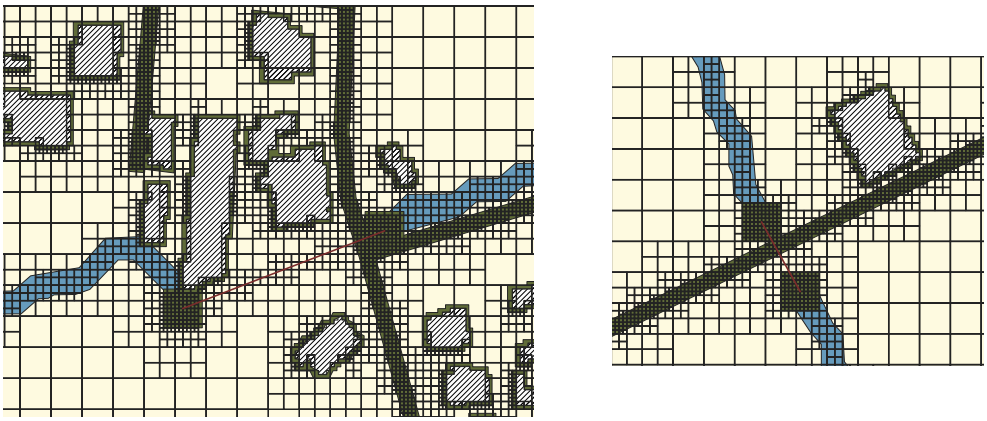


Figure 2 Quadtree-refined meshes. Base resolution is 8 m, with refinements down to 1 m around streets, culverts, buildings (green) and 2 m around streams (blue). Culverts are marked in red.

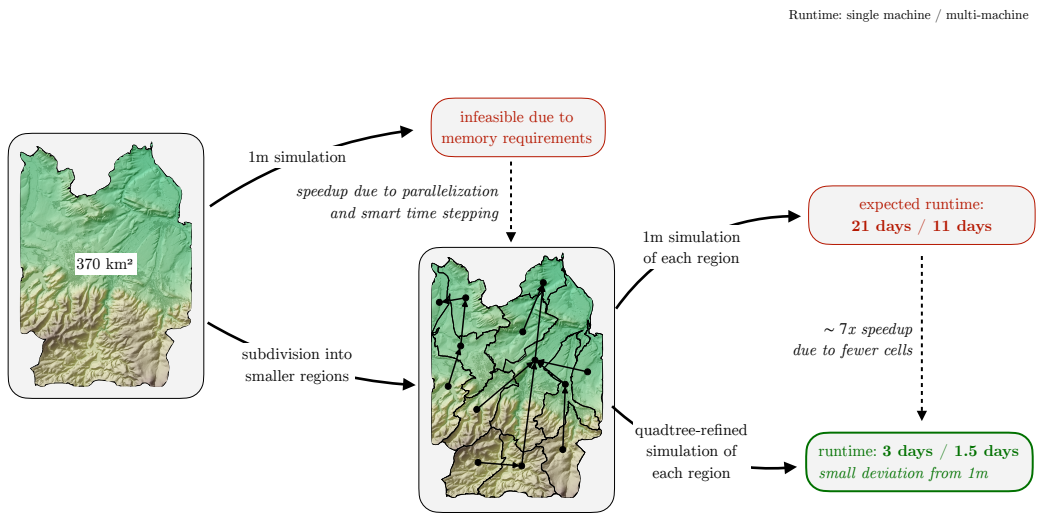


Figure 3 Overview of the interplay between our techniques for simulating large areas with high resolution and their trade-offs in terms of runtimes.

many factors, such as the amount of precipitation, condition numbers, terrain characteristics and the presence of 1D model elements such as culverts, which are also modeled. For very large domains (such as our area of interest of 370 km^2 in the mountainous Harz region) at 1 m resolution, the limitations of available GPU memory prohibits running a simulation on the entire region at once. Our subdivision technique makes such a simulation feasible, requiring roughly three weeks to simulate a five-hour heavy-rain event on a single machine and eleven days when parallelization is used to the full extent possible with the given dependency graph. Combining this with our grid refinement technique reduces the required time to three days on a single machine and to around 36 hours when parallelized, see Figure 3.

This progress enables us to demonstrate how the second challenge can be addressed: (II) We use our new computational capabilities to conduct a large-scale, multifaceted sensitivity analysis, providing insights into different scenarios for possible time-dependent runoff and flooding when varying the input precipitation for our study area. This makes it possible to identify both regions of stability (for which the outcomes are largely unchanged under input variations) as well as critically unstable locations (for which the outcome may change significantly under input variations). In particular the latter is of crucial value for effective disaster mitigation: Knowing where relatively small changes can make a significant difference in the outcome is highly valuable when deciding where to invest limited resources with the greatest impact. We demonstrate the viability of this approach by identifying such a critical location for our study area, see Figure 4.

5 The Video

After introducing the global challenges and previous work from Computational Geometry (culminating in SCALGO), the video describes the challenges and approaches to dealing with *time-dependet, dynamic* scenarios, such as in our study region in the Harz mountains. After visualizing the two main outcomes from above, we highlight how these improvements make it possible to identify critical locations, where small changes can have a large effect.

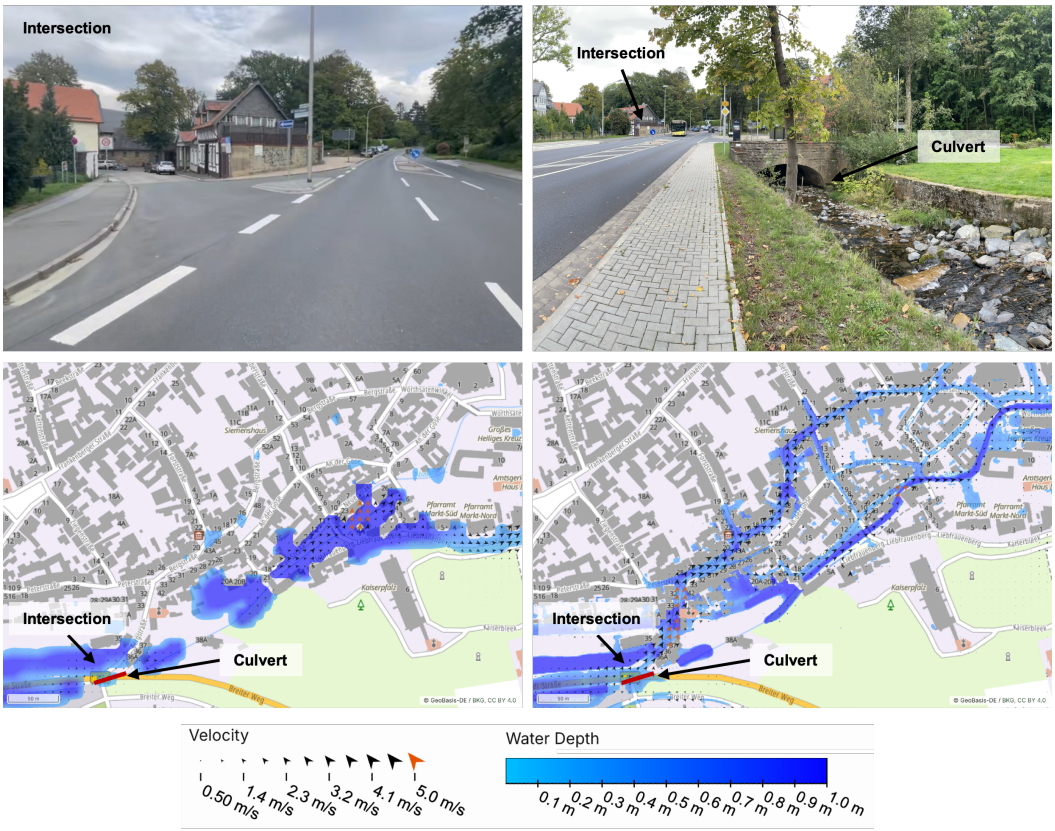


Figure 4 A critical location near Goslar, identified by further sensitivity analysis. (Top left) Close-up of the intersection. (Top right) A culvert that runs below the intersection. (Bottom) Criticality of this intersection: maximum water depth and velocity from uniform-resolution simulations.

References

1 Pankaj K. Agarwal, Lars Arge, Thomas Mølhave, and Bardia Sadri. I/O-efficient efficient algorithms for computing contours on a terrain. In *Symposium on Computational Geometry (SoCG)*, pages 129–138, 2008. doi:10.1145/1377676.1377698.

2 Pankaj K. Agarwal, Lars Arge, and Ke Yi. I/O-efficient batched union-find and its applications to terrain analysis. In *Symposium on Computational Geometry (SoCG)*, pages 167–176, 2006. doi:10.1145/1137856.1137884.

3 Pankaj K. Agarwal, Lars Arge, and Ke Yi. I/O-efficient batched union-find and its applications to terrain analysis. *ACM Transactions on Algorithms*, 7(1):11:1–11:21, 2010. doi:10.1145/1868237.1868249.

4 Lars Arge, Aaron Lowe, Svend C. Svendsen, and Pankaj K. Agarwal. 1D and 2D flow routing on a terrain. *ACM Transactions on Spatial Algorithms and Systems*, 9(1):3:1–3:39, 2023. doi:10.1145/3539660.

5 Lars Arge and Morten Revsbæk. I/o-efficient contour tree simplification. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 1155–1165, 2009. doi:10.1007/978-3-642-10631-6_116.

6 Lars Arge, Morten Revsbaek, and Norbert Zeh. I/O-efficient computation of water flow across a terrain. In *Symposium on Computational Geometry (SoCG)*, pages 403–412, 2010. doi:10.1145/1810959.1811026.

7 Lars Arge and Norbert Zeh. I/O-efficient strong connectivity and depth-first search for directed planar graphs. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, page 261, 2003. doi:10.1109/SFCS.2003.1238200.

8 Kate Bradbrook, Simon Waller, and Dave Morris. National floodplain mapping: datasets and methods–160,000 km in 12 months. *Natural hazards*, 36:103–123, 2005. doi:10.1007/s11069-004-4544-9.

9 Copernicus Digital Elevation Model (DEM). Global 30 m, version 1.1. https://registry.opendata.aws/copernicus-dem/, 2021. European Space Agency & European Commission (accessed 2025-06-23).

10 Detlef Deumlich and Andreas Gericke. Frequency trend analysis of heavy rainfall days for germany. *Water*, 12(7):1950, 2020. doi:10.3390/w12071950.

11 RJ Hardy, PD Bates, and MG Anderson. The importance of spatial resolution in hydraulic models for floodplain environments. *Journal of Hydrology*, 216(1-2):124–136, 1999. doi:10.1016/S0022-1694(99)00002-5.

12 Neil M Hunter, PD Bates, S Neelz, G Pender, I Villanueva, NG Wright, D Liang, Roger Alexander Falconer, B Lin, S Waller, et al. Benchmarking 2d hydraulic models for urban flooding. In *Proceedings of the institution of civil engineers-water management*, volume 161, pages 13–30, 2008. doi:10.1680/wama.2008.161.1.13.

13 Shubham M Jibhakate, PV Timbadiya, and PL Patel. Flood hazard assessment for the coastal urban floodplain using 1d/2d coupled hydrodynamic model. *Natural Hazards*, 116(2):1557–1590, 2023. doi:10.1007/s11069-022-05728-7.

14 Thomas Mølhave, Pankaj K. Agarwal, Lars Arge, and Morten Revsbæk. Scalable algorithms for large high-resolution terrain data. In *International Conference and Exhibition on Computing for Geospatial Research & Application (COM.Geo)*, pages 20:1–20:21, 2010. doi:10.1145/1823854.1823878.

15 Michael Perk, Phillip Keldenich, Tobias Wallner, and Sándor P Fekete. Hydrodynamic flows during heavy rain events: Scalable algorithmic methods for simulation and sensitivity analysis. *EGUsphere*, pages 1–33, 2026.

16 TUFLOW. Tuflow user manual, 2025. URL: https://docs.tuflow.com/.

17 United Nations Office for Disaster Risk Reduction. *Global assessment report on disaster risk reduction 2025: Resilience Pays: Financing and Investing for our Future*. UN, 2025. URL: https://www.undrr.org/gar/gar2025.

Interactive Visualization and Verification Tools for Tesseract Path Unfoldings

Soham Samanta
Greater Commonwealth Virtual School, Medford, MA, USA

Hugo A. Akitaya
University of Massachusetts Lowell, MA, USA

Erik Demaine
MIT CSAIL, Cambridge, MA, USA

Martin Demaine
MIT CSAIL, Cambridge, MA, USA

Abstract

This paper introduces interactive software tools for studying 2-face path unfoldings of the tesseract (4D hypercube). We present: (1) an algorithm to verify whether a given 24-omino is a valid path unfolding of the tesseract, (2) a web-based visualization tool for exploring and animating unfolding sequences with smooth 3D interpolation, and (3) a design interface integrated with SVG Painter, with a similar design to Demaine's SVG Painter [3], to create custom unfoldings. We demonstrate these tools by designing a geometric font of 36 path unfoldings resembling Latin letters and digits, illustrating the rich diversity and accessibility of tesseract geometry.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases unfolding, polyominoes, tesseract, path unfolding, visualization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.104

Category Media Exposition

Supplementary Material *Software (Web Application)*:
`https://github.com/Soham2020sam/PathUnfoldingsTesseractVisualizationandFONT`
archived at `swh:1:dir:9f0e070a9bebcdd3c03d5f0035788345db6b510d`

Funding *Hugo A. Akitaya*: Supported by the NSF award CCF-2348067.

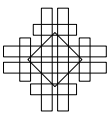
1 Introduction

Unfolding polyhedra is a classical problem in discrete and computational geometry [4]. In the traditional *edge unfolding* problem, we cut a subset of edges of a given polyhedron and unfold its surface so that the faces remain connected and lie flat in the plane without overlap. A natural extension to higher dimensions is *ridge unfolding*: cut the $(d-2)$-dimensional faces (ridges) of a d-dimensional polytope and lay down the $(d-1)$-dimensional faces (facets) flat in the plane. Turney [8] showed that there are exactly 261 polycube unfoldings of the tesseract (4D hypercube), and Diaz and O'Rourke [5] investigated the 2D polyomino unfoldings derived from these 3D unfoldings.

Recently, Akitaya and Kandarpa [1] introduced *2-face unfoldings*, which unfolds the 2-skeleton of a polytope directly to 2D. The *2-skeleton* consists of all 2-dimensional faces glued together at their shared edges. A 2-face unfolding cuts some of these shared edges so that the 2-skeleton unfolds to a connected, nonoverlapping 2D polyomino. For the tesseract, all 24 square faces appear exactly once, yielding a 24-omino.

In this paper, we focus on *path unfoldings* where the dual graph forms a Hamiltonian path through the 24 square faces. These have the maximum possible diameter in the dual graph. The tesseract has eight 3D cube faces, including four pairs of opposite cubes that


© Soham Samanta, Hugo A. Akitaya, Erik Demaine, and Martin Demaine;
licensed under Creative Commons License CC-BY 4.0



share no square face. We call one such opposite pair the *top* and *bottom* cubes. The twelve remaining square faces form the *side faces* connecting top and bottom. In prior work [2, 7], a subset of the authors studied *alternating* path unfoldings – those that visit top and bottom cubes alternately (six times each) with a side face visit in between – and enumerated all 35,520 such unfoldings.

However, working with tesseract unfoldings presents significant challenges: understanding the 4D geometry requires substantial mental effort, verifying whether a proposed polyomino can fold into the tesseract is non-trivial, and designing specific unfolding shapes requires iterative trial and error. These difficulties motivated the development of computational tools to make tesseract unfoldings more accessible to researchers.

Contribution. This paper introduces software tools to facilitate research on tesseract path unfoldings: (1) A **verification algorithm** that checks whether a given 24-omino is a valid path unfolding using recursive backtracking; (2) A **web-based visualization tool** that renders unfoldings and animates folding sequences with smooth 3D interpolation; (3) An integrated **design workflow** combining SVG Painter with our verification tool for rapid prototyping.

2 Software

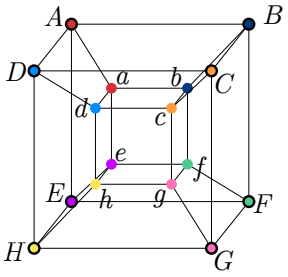


Figure 1 Naming convention for vertices. Top cube: $ABCDabcd$; bottom cube: $EFGHefgh$; side faces connect them.

Tesseract Structure. We label the top cube as $ABCDabcd$ and the bottom cube as $EFGHefgh$. The twelve side faces connect the top and bottom cubes. This labeling convention is used throughout our visualization tool, as in Figure 1.

Path Unfolding Checker. We developed a verification algorithm that determines whether a given 24-omino can be folded into the 2-skeleton of the tesseract via a Hamiltonian path. The algorithm uses recursive backtracking to simultaneously: traverse a potential Hamiltonian path in the dual graph of the input polyomino, walk through the face-adjacency graph of the tesseract, and verify that adjacencies in the polyomino correspond to valid face adjacencies in the tesseract.

The algorithm maintains a partial mapping from polyomino squares to tesseract faces. At each step, it selects an unmapped square adjacent to the current path endpoint, tries assigning it to each tesseract face that could validly extend the path, and recursively attempts to complete the mapping or backtracks if no valid completion exists.

Complexity. The search space is bounded by the number of tesseract faces (24) times the branching factor at each step, which is at most the degree in the face-adjacency graph of the tesseract (at most 4 neighbors per square face). In the worst case (an invalid input requiring full exploration), the algorithm explores $O(4^{24})$ partial assignments, though in practice aggressive pruning via adjacency constraints reduces this dramatically. On all inputs tested, including invalid 24-ominoes, verification completes in well under one second in a browser environment.

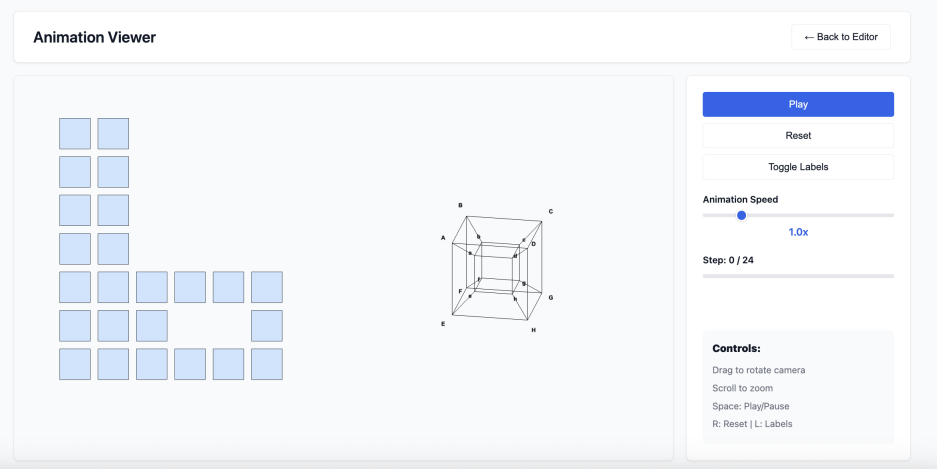


Figure 2 Web-based visualization tool interface showing the 24-omino with face labels and animation controls.

Visualization Tool. Our web-based visualization tool provides an interactive interface for exploring tesseract path unfoldings. Built using HTML5 Canvas and Three.js, the tool features: face labeling, color schemes highlighting cube structure, dual path visualization, export options, interactive zoom/pan controls, and 3D folding animation.

Animation Mathematics. The visualization animates how the flat 24-omino folds into the tesseract's 2-skeleton using smooth interpolation between the 2D unfolding (source) and 3D tesseract projection (destination).

Smoothstep Interpolation. For visually pleasing animations with smooth acceleration and deceleration, we use Perlin's smoothstep function [6]. Given linear time parameter $t \in [0, 1]$, we compute smoothed parameter $s(t) = 6t^5 - 15t^4 + 10t^3$. This quintic polynomial satisfies $s(0) = 0$, $s(1) = 1$, with $s'(0) = s'(1) = 0$ and $s''(0) = s''(1) = 0$, creating smooth ease-in/ease-out.

Position and Vertex Interpolation. For each face i, we compute source position $\mathbf{p}_{\text{src}}$ (center in flat unfolding) and destination position $\mathbf{p}_{\text{dst}}$ (center in tesseract projection). The animated center position is $\mathbf{p}(t) = \mathbf{p}_{\text{src}} + (\mathbf{p}_{\text{dst}} - \mathbf{p}_{\text{src}}) \cdot s(t)$. Each vertex j similarly interpolates: $\mathbf{v}_j(t) = \mathbf{v}_{\text{src},j} + (\mathbf{v}_{\text{dst},j} - \mathbf{v}_{\text{src},j}) \cdot s(t)$, morphing the square from flat to its 3D orientation.

Rotation and Projection. The tesseract projection is interactively rotatable via mouse drag. We apply rotation matrices: first y-axis rotation by θ_y, then x-axis rotation by θ_x:

$$R_y(\theta_y) = \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{pmatrix}, \quad R_x(\theta_x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix} \tag{1}$$

We use orthographic projection (discarding z after transformations) to preserve parallel lines and geometric relationships, avoiding perspective distortion.

The animation sequence proceeds face-by-face: each face turns white, interpolates from flat position to tesseract over $t \in [0, 1]$, then joins the tesseract structure. Animation speed is user-controllable, and the tesseract can be rotated during animation.

Design Workflow. To create custom path unfoldings, we developed an integrated workflow: (1) Sketch 24-omino in SVG Painter in our format. (2) Verify using our checker algorithm; (3) Iterate if invalid; (4) Visualize and animate the valid unfolding.

We created our FONT using this method, which can be seen below.

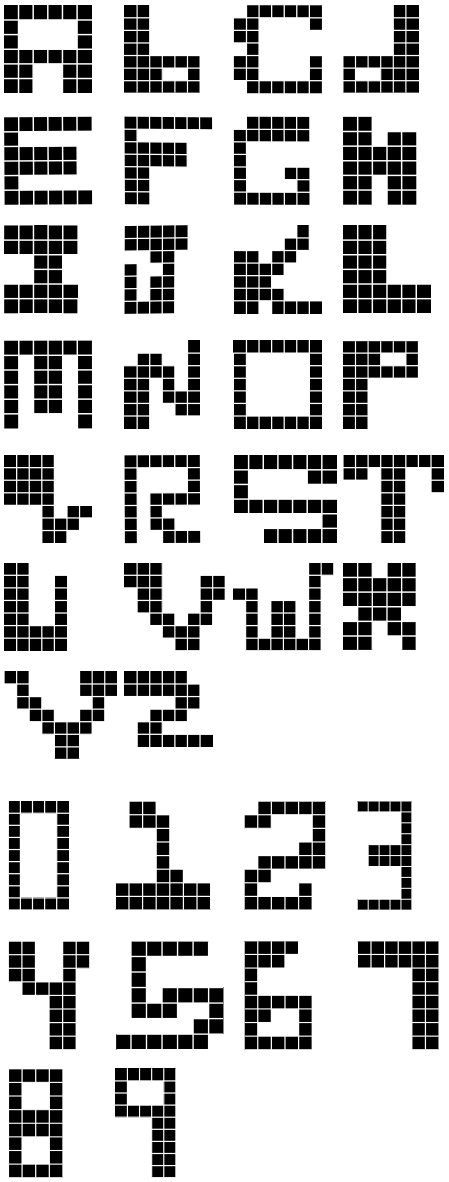


Figure 3 Tesseract font. Each letter/digit is represented by a path 2-face unfolding of the tesseract.

References

1 Hugo A. Akitaya and Nithish V. Kandarpa. Unfolding skeletons. In *Proc. 24th Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCGGG)*, 2022.

2 Hugo A. Akitaya and Soham Samanta. Path-unfolding the tesseract. In *Proc. 31st Fall Workshop on Computational Geometry*, 2024.

3 Erik D. Demaine. SVG Painter. URL: `https://erikdemaine.org/svgpainter/`.

4 Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

5 Giovanna Diaz and Joseph O'Rourke. Hypercube unfoldings that tile $\mathbb{R}^3$ and $\mathbb{R}^2$, 2015. `arXiv:1512.02086`.

6 Ken Perlin. Improving noise. *ACM Transactions on Graphics*, 21(3):681–682, July 2002. `doi:10.1145/566654.566636`.

7 Soham Samanta, Hugo Akitaya, Erik Demaine, and Martin Demaine. 2-face path unfolding the tesseract into a font. In *Proc. 32nd Fall Workshop on Computational Geometry*, 2025. To appear.

8 Peter D. Turney. Unfolding the tesseract. *Journal of Recreational Mathematics*, 17(1):1–16, 1984.

ETH Flippers Approach to Parallel Reconfiguration of Triangulations: SAT Formulation and Heuristics

Lorenzo Battini
ETH Zürich, Switzerland

Marko Milenković
ETH Zürich, Switzerland

Abstract

We describe the algorithms used by the ETH Flippers team in the CG:SHOP 2026 Challenge. Each instance consists of a set of triangulations on a common point set, and the objective is to find a central triangulation that minimizes the total parallel flip distance to the input set. Our strategy combines an exact solver for small and medium-sized instances with a suite of heuristics for larger instances. For the exact approach, we formulate the problem as a SAT instance with XOR clauses to model edge transitions across multiple rounds, further optimized by lower bounds derived from exact pairwise distances. For larger instances, we use a greedy local search and edge-coloring techniques to identify maximal sets of independent flips. Our approach ranked second overall and first in the junior category, computing provably optimal solutions for 186 out of 250 instances.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases exact solution, heuristic, SAT solver, XOR clauses, computational geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.105

Category CG Challenge

Related Version *Extended Version*: `https://arxiv.org/abs/2603.22456`

Supplementary Material *Software*: `https://github.com/Lbattini/ETH-flippers-CGSHOP2026` [2] archived at `swh:1:dir:ec12f1b858c98832fd57fc7a2d407989c1fc00cf`

Acknowledgements We would like to thank the Challenge organizers and other competitors for their time, feedback, and for making this whole event possible. This work was conducted as part of the Practical Work course at ETH Zürich under the supervision of Michael Hoffmann, whose guidance was invaluable. We sincerely thank Bernd Gärtner for providing travel support to attend the conference. We also acknowledge the ETH High Performance Computing group for providing the computational resources of the Euler cluster.

1 Introduction

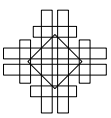
Triangulations of point sets, in this context, maximal straight-line crossing-free graphs with endpoints in the point set, have many applications, for instance, in computer graphics and in reconstruction problems in geodesy.

A common problem is transforming a triangulation into another on the same point set using edge flips, replacing an internal edge whose incident triangles form a convex quadrilateral with its other diagonal. Hurtado, Noy, and Urrutia [4] introduced parallel edge flips, which consist of flipping at the same time a set of edges such that none of the pairs of edges in the set shares a triangle.

The "CG:SHOP Challenge" (Computational Geometry: Solving Hard Optimization Problems) is a computational challenge competition about a hard geometric optimization problem. This year's challenge problem was proposed by Aichholzer, Dorfer, and Kramer. The goal is to find a central triangulation C and, for each of the given m triangulations



© Lorenzo Battini and Marko Milenković;
licensed under Creative Commons License CC-BY 4.0



$T_1, T_2, \ldots, T_m$ on the same pointset P, a sequence of parallel flips transforming T_i into C, such that the sum of the lengths of these sequences is minimized. The solution methods are evaluated on a benchmark of 250 instances. We refer to [1] for further challenge details.

We present an exact solution method based on formulating the problem as a SAT problem with XOR clauses, which is used to solve the small and medium instances, as well as various heuristics that we used for the larger instances.

Our team, ETH flippers, ranked second overall and first in the junior category. We computed a provably optimal solution for 186 instances. Three further instances were solved, but a technical bug during our local validation caused us to stop at sub-optimal values. The largest instances we optimally solved featured twenty 320-point triangulations, while the highest total parallel flip distance among our optimally solved instances is 92. Notably, for this latter instance (`random_instance_440_160_20`), our solution was uniquely the best among all participating teams (see Figure 1).

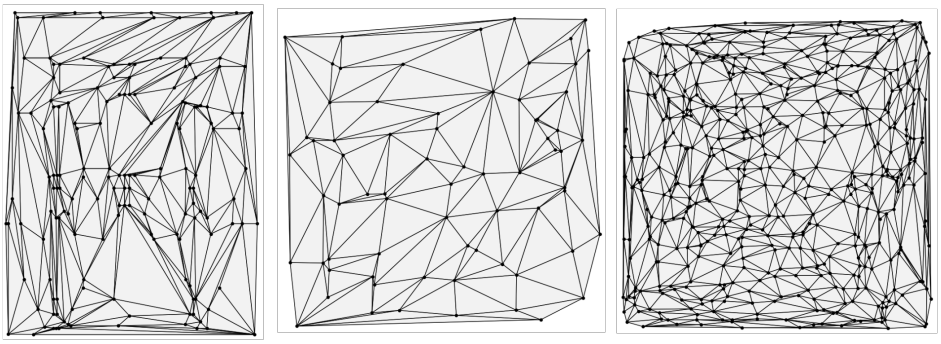


Figure 1 Our best centers to instances `random_instance_440_160_20`, `woc-70-random-9a7d18d3`, `rirs-500-50-23d00ec5`.

2 2-SAT formulation

Given a set of m triangulations, $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$, we formulate a 2-SAT instance to search for a triangulation C such that the distance from each $T_i \in \mathcal{T}$ to C is at most one.

For every convex quadrilateral q formed by a pair of facial triangles that share an edge in any T_i, we fix an arbitrary reference diagonal. Let the variable x_q denote whether C uses its reference diagonal and let the constant $b_q^{(i)}$ denote its state in T_i. To prevent invalid simultaneous flips, we add the clause $\neg(b_q^{(i)} \oplus x_q) \vee \neg(b_{q'}^{(i)} \oplus x_{q'})$ for all pairs of intersecting quadrilaterals q, q' in each T_i. This determines the existence of a central triangulation C in $\mathcal{O}(mn)$ time.

3 SAT formulation with XOR clauses

XOR clauses. The most natural approach to modelling parallel flips is a round-by-round progression. We need to keep the state of each edge across intermediate parallel flips.

We model the state transition of an edge e over k parallel flips using XOR operations. Let $e^0, e^k \in \{0, 1\}$ denote the presence of e in the initial and final triangulations, respectively. Let $\{q_1, \ldots, q_g\}$ be the set of all potential convex empty quadrilaterals that share e as a diagonal. We let $f_i^j \in \{0, 1\}$ indicate whether a flip is performed on q_i during step j.

Because the final state of edge e after k parallel flips is simply its initial state XORed with all intermediate flips across its potential quadrilaterals $q_{1\dots g}$, we add the constraint

$$e^0 \oplus e^k \oplus \bigoplus_{j=1}^{k} \bigoplus_{i=1}^{g} f_i^j = 0$$

We decided to use the CryptoMiniSat5 [8] solver, since it supports inputs with XOR clauses.

Formulation

We extend the previous formulation to search for a center C whose distance to each $T_i \in \mathcal{T}$ is upper bounded by a given target $d[i]$.

Geometric Preconditions for Flips. Let $\mathcal{Q}$ be the set of all convex empty quadrilaterals defined on the pointset, which we identified via spatial hashing [9]. Let the variable e_{uv}^t denote the presence of edge (u, v) at step t. For a flip $f_q^t \in \{0, 1\}$ to be geometrically valid at step t, the four boundary edges ∂q of its corresponding quadrilateral $q \in \mathcal{Q}$ must exist in the prior step. We enforce this constraint with implication $f_q^t \implies \bigwedge_{(u,v)\in\partial q} e_{uv}^{t-1}$.

Target Equivalence with Center C. We introduce global boolean variables c_{uv} representing the edges of the center C. We enforce consistency at the final step k via the equivalence $e_{uv}^k \iff c_{uv}$. This link forces the final state of every flip sequence to be identical to the global center, effectively merging the independent SAT sub-problems to a shared solution.

Edge Transition Constraints. We must ensure that the triangulation changes only through valid flip operations. An edge (u, v) changes its status (present or absent) between parallel flips $t-1$ and t if and only if a flip is performed on a quadrilateral where (u, v) serves as the diagonal.

Let $\mathcal{Q}_{uv}$ be the set of all convex empty quadrilaterals for which (u, v) is a diagonal. The relationship between the edge's state at consecutive parallel flips and the flip variables is enforced via an XOR constraint: $e_{uv}^{t-1} \oplus e_{uv}^t \oplus \bigoplus_{q\in\mathcal{Q}_{uv}} f_q^t = 0$.

For edges on the convex hull, the set $\mathcal{Q}_{uv}$ is empty, effectively enforcing that these edges remain constant throughout the transformation.

Constraints on Simultaneous Flips. We must enforce strict independence between simultaneous operations. Specifically, for any subset of four vertices $\{v_1, v_2, v_3, v_4\}$, there can be at most one active flip operation at time t involving three of these vertices. This prevents structural conflicts where multiple flips compete for the same triangle.

We iterate through all quadruplets of vertices and identify the set of potential flip variables $\mathcal{F}_{v_{1..4}}^t$ of size r associated with them. We then enforce an *At-Most-One* (AMO) constraint on this set $\sum_{f\in\mathcal{F}_{v_{1..4}}^t} f \leq 1$. There are multiple ways to model this constraint using a linear number of clauses [7]. The ladder encoding that we implemented uses $2r-1$ variables and $3r-2$ clauses, compared to r variables and $\binom{r}{2}$ clauses. Due to the $\mathcal{O}(n^4)$ possible number of convex empty quadrilaterals, clauses generation becomes a computational bottleneck.

Initial State and Boundary Conditions. We initialize the SAT model for each input triangulation by fixing e_{uv}^0 to true if the edge (u, v) exists in it and false otherwise. Furthermore, since parallel flips only affect internal diagonals of convex quadrilaterals, the convex hull edges $\mathcal{H}$ remain invariant. We hard-code e_{uv}^t to true for all $(u, v) \in \mathcal{H}$ across all steps $t \in \{1, \dots, k\}$, effectively pruning the search space by preventing flips of boundary edges.

Pruning clauses. To significantly prune the search space, we precompute the minimum flip distance required to introduce each edge into triangulation T_i using an iterative breadth-first reachability search. We then omit all SAT variables and clauses associated with an edge for any time step prior to this theoretical minimum.

4 Exact solving

Lower bound. To establish a lower bound, we first compute exact pairwise parallel flip distances $D_{i,j}$ using our SAT formulation. Any valid central triangulation at distances d_k to inputs T_k must satisfy the triangle inequality $d_i + d_j \geq D_{i,j}$ for all pairs i, j. To test if a target total distance S is feasible, we find all integer assignments $(d_1, \ldots, d_m)$ summing to S that satisfy these pairwise constraints via recursive backtracking. For instances with $m = 20$ triangulations, we use some further pruning techniques. We compute the minimum parallel flip distance $s_{dist}(i)$ for the last i triangulations, for $i \geq i_{start}$, where we used $i_{start} = 11$ for most instances since it was experimentally found to be sufficiently fast to compute. Then we use those suffix distances by pruning a branch whenever $S - \sum_{k=1}^{i} d_k < s_{dist}(i)$. Moreover, after the first 10 and 15 distances have been assigned recursively, we prune the branch if the prefix is not satisfiable.

Choosing the distribution of distances. The upper bound search is existential. Consequently, prioritizing candidates that are more likely to be satisfiable can significantly accelerate the search.

Let $d = (d_1, \ldots, d_n)$ represent a potential assignment vector. We define the scoring function $\mathcal{S}(d)$ as the sum of absolute differences between all pairs of elements: $\mathcal{S}(d) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |d_i - d_j|$. The candidate distributions are processed in ascending order of $\mathcal{S}(d)$.

Empirical analysis reveals that this heuristic is remarkably effective for randomly generated instances. In the majority of cases, the satisfiable distribution corresponds to the candidate with the minimal $\mathcal{S}(d)$ score. Even in cases where the global minimum is not the solution, the valid assignment is consistently found among the top-ranked candidates.

▶ **Conjecture 1.** *For a randomly generated triangulation instance under the standard uniform distribution model, ordering of candidate distance vectors by increasing $\mathcal{S}(x)$ minimizes the expected number of queries required to find a satisfiable assignment.*

5 Heuristics

Transforming a triangulation into another. To transform a triangulation T into another T', we flip edges such that the number of intersecting edges decreases. We now describe how, at each step, we choose a subset of independent edges to flip among the ones that reduce the number of intersections.

Firstly, we choose edges that, if flipped, introduce an edge that is present in T'. To do so, we start by creating a graph containing those edges, where the nodes correspond to faces incident to them. Then we compute an edge coloring of this graph using the Misra and Gries algorithm [5] from the Boost Graph Library [6]. Finally, we flip the edges corresponding to the color class of maximum cardinality, and all the isolated edges. Since the boundary edges cannot be flipped, and the bounded faces are triangles, the maximum degree Δ is at most 3. Since the Misra and Gries algorithm uses at most $\Delta + 1 \leq 4$ colors, at least one fourth of the edges in the graph are flipped.

Secondly, we greedily pick the edges that are independent from all the edges that have already been picked, in order of non-increasing reduction of the number of intersections.

This transforms a triangulation into the other in a finite number of steps, since it can be seen as an adaptation to parallel flips of the algorithm described by Hanke, Ottmann, and Schuierer [3]. Given the flip sequence $\mathcal{F} = \{F_1, F_2, ..., F_{|\mathcal{F}|-1}\}$ that we obtain with the described heuristic, let $\{\tau_0 = T, \tau_1, ..., \tau_{|\mathcal{F}|-1} = T'\}$ be the corresponding sequence of triangulations, where τ_{k+1} is the triangulation we obtain after the parallel flip F_k. The reversed flip sequence $\mathcal{F}'$ transforms T' into T and it can be obtained by reversing the order of the parallel flips, and replacing each individual flip $e \to e'$ with $e' \to e$.

We can obtain other heuristic flip sequences $\mathcal{F}_k$, with $k \in [0, |\mathcal{F}| - 1]$, by concatenating the heuristic flip sequence that turns T into τ_k with the reversed flip sequence that turns T' into τ_k.

Since these sequences are heuristic, their length might differ, and so we try multiple values of k and choose the shortest flip sequence.

We empirically observed that trying all intermediate k values typically reduces each triangulation's distance by 1. Therefore, to maintain computational efficiency on large instances, we only do this when the total distance is within $currentBest + m$. Here, $currentBest$ is the smallest objective value among all the centers that were computed before.

Choosing centers. Apart from input triangulations, we also generate candidate centers by greedily flipping edges in input triangulations to minimize total intersections with the entire point set until a local optimum is reached.

We then record all the intermediate triangulations at the end of each parallel flip, which is used to transform an input triangulation into one of those centers. To avoid trying all centers, we sort them in increasing order of total number of intersections with all the input triangulations and only compute the heuristic parallel flip distance for a fraction of them.

6 Combined Strategies

Fixed center. A promising combined strategy involves using the heuristic to select a candidate center triangulation, denoted as T_{heur}, and using the exact SAT solver to compute the minimal parallel flip distances from all input triangulations to T_{heur}, one by one.

Although this approach remains heuristic, it was effective on the *rirs* instances, but it also indirectly helped significantly with large non-*rirs* instances. Finding a slightly better solution allows the exact solver to handle cases that were previously out of reach.

Solving a subproblem. A natural extension to computing pairwise distances is to compute exact distances for larger subsets. This yielded negligible improvements to the lower bound, likely due to the triangle inequality holding strong even for pairwise constraints.

Similar reasoning proved highly effective for the *woc* (especially *tsplib*) instances, observing that the distances to the optimal center were not uniformly distributed. By identifying and excluding outlier triangulations, we could exactly solve for the center of the remaining subset. This produced a robust heuristic center for the fixed center heuristic.

References

1 Oswin Aichholzer, Joseph Dorfer, Sándor P. Fekete, Phillip Keldenich, Peter Kramer, and Stefan Schirra. Central Triangulation under Parallel Flip Operations: The CG:SHOP Challenge 2026, 2026. arXiv:2603.18812.

2 Lorenzo Battini and Marko Milenković. ETH flippers CGSHOP2026. Software, swhId: swh:1:dir:ec12f1b858c98832fd57fc7a2d407989c1fc00cf (visited on 2026-05-15). URL: https://github.com/Lbattini/ETH-flippers-CGSHOP2026, doi:10.4230/artifacts.26084.

3 Sabine Hanke, Thomas Ottmann, and Sven Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2, April 1996. doi:10.3217/jucs-002-08-0570.

4 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Parallel edge flipping. In *CCCG*, 1998.

5 J. Misra and David Gries. A constructive proof of Vizing's theorem. *Information Processing Letters*, 41(3):131–133, 1992. doi:10.1016/0020-0190(92)90041-S.

6 Maciej Piechotka. Edge coloring BGL implementation. URL: https://www.boost.org/doc/libs/latest/libs/graph/doc/edge_coloring.html.

7 Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *Principles and Practice of Constraint Programming – CP 2005*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005. doi:10.1007/11564751_73.

8 Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to Cryptographic problems. In *Theory and Applications of Satisfiability Testing – SAT 2009*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2009. doi:10.1007/978-3-642-02777-2_24.

9 Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomeranets, and Markus Gross. Optimized spatial hashing for collision detection of deformable objects. *Proceedings of Vision, Modeling, and Visualization*, 3:47–54, 2003.

Engineering Greedy Heuristics and Simulated Annealing Methods for the Median Triangulation Under the Parallel Flip Distance

Jacobus Conradi
BARC, University of Copenhagen, Denmark

Benedikt Kolbe
Hausdorff Center for Mathematics, University of Bonn, Germany

Philip Mayer[1]
Institute of Computer Science, University of Bonn, Germany

Jonas Sauer
Karlsruhe Institute of Technology, Germany

Jack Spalding-Jamieson
Independent, Vancouver, Canada

Abstract

We present our approach for the CG:SHOP 2026 challenge. In this international challenge, the goal was to find a median triangulation for a set of triangulations in the parallel flip reconfiguration graph of all triangulations of an underlying point set. Our simulated-annealing-based approach makes use of two ingredients: a heuristic edge selection for approximating the parallel flip distance of two given triangulations, and a heuristic procedure to generate good initial triangulations.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases triangulation, flip distance, parallel flip distance, heuristic, competition

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.106

Category CG Challenge

Supplementary Material *Software (Source Code)*: `https://github.com/PhilipMayer94/jflip` archived at `swh:1:dir:5f40dddfbe834cfc16e09dcdac08d2c61d979042`

Funding *Jacobus Conradi*: Supported by the Carlsberg Foundation, grant CF24-1929.
Benedikt Kolbe: This work was partially supported by the Lamarr Institute for Machine Learning and Artificial Intelligence.
Philip Mayer: Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant FOR-5361 – 459420781.
Jonas Sauer: Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant FOR-5361 – 459420781.

Acknowledgements We thank the challenge organizers and other competitors for their time, feedback, and for making this competition and this research possible. We gratefully acknowledge access to the Marvin cluster of the University of Bonn.

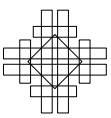
1 Introduction

In this paper, we study changing triangulations of a planar point set using *parallel flips*. Given a set of planar triangulations of some point set, we search for a central triangulation and a sequence of parallel flip operations from each input triangulation to the central triangulation, so as to minimize the sum of the number of parallel flip operations in each sequence.

[1] corresponding author



© Jacobus Conradi, Benedikt Kolbe, Philip Mayer, Jonas Sauer, and Jack Spalding-Jamieson;
licensed under Creative Commons License CC-BY 4.0



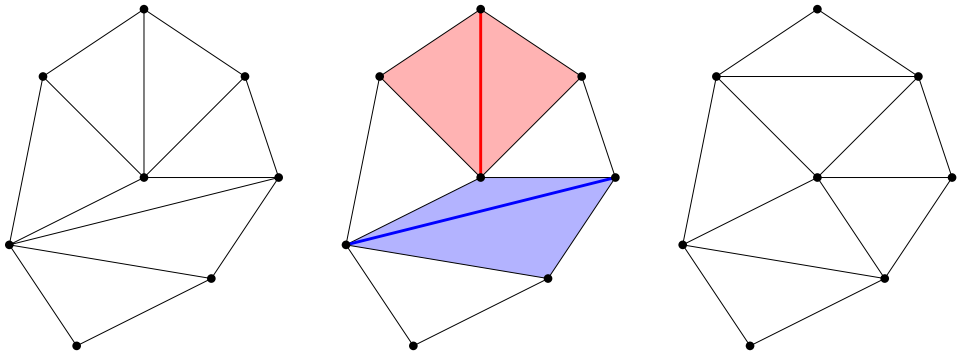


Figure 1 Two edges with their corresponding quadrilateral. They do not share a triangle and can be flipped in parallel, transforming the left triangulation into the right triangulation.

This problem represents the core of the 2026 Computational Geometry Challenge [1] (CG:SHOP), which ranked teams based on their solutions to a provided set of instances. Our team, `jflip2`, was invited to publish its methods after finishing in 4th place with an approach based on heuristic methods inspired by observations for the classical flip distance [5].

For a point set $\mathcal{S} \subset \mathbb{R}^2$, a triangulation $\mathcal{D}$ is a maximal set of non-crossing straight-line edges incident to points in $\mathcal{S}$. In a triangulation, every interior edge e not on the boundary of $\text{conv}(\mathcal{S})$ is adjacent to exactly two triangles. If two such triangles form a convex quadrilateral Q, then we call e *flippable*, and replacing e with the other diagonal $\text{flip}(e)$ results in a valid triangulation $\mathcal{D}' = \mathcal{D} \setminus \{e\} \cup \{\text{flip}(e)\}$. Replacing e with $\text{flip}(e)$ is called a *unit flip*.

A set of edges E in a triangulation $\mathcal{D}$ is called a *parallel flip operation* if each edge $e \in E$ is flippable and no two edges in E are part of the same triangle [4]. Since no two edges in a parallel flip share a triangle, we again obtain that $\mathcal{D}' = \mathcal{D} \setminus E \cup \{\text{flip}(e) \mid e \in E\}$ is a valid triangulation. Parallel flips induce a graph PG, whose vertices are the set $\mathfrak{D}$ of all triangulations, in which two triangulations are adjacent if one can be transformed into the other by exactly one parallel flip. Note that the edges of this graph are undirected, since parallel flips are reversible, i.e., $\text{flip}(\text{flip}(e)) = e$, and, classically, this graph is connected [7]. A sequence of parallel flip operations transforming $\mathcal{D}$ to $\mathcal{D}'$ is a path in PG from $\mathcal{D}$ to $\mathcal{D}'$. The length $\mathrm{d}_{\text{pf}}(\mathcal{D}, \mathcal{D}')$ of a shortest path from $\mathcal{D}$ to $\mathcal{D}'$ in PG is called the *parallel flip distance*.

Given these definitions, we formulate the challenge posed in the CG:SHOP contest.

▶ **Problem 1** (Median Point for Parallel Flip Distance). *Given a set $\mathcal{S}$ of points and a set $\mathfrak{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$ of triangulations (terminals), find a triangulation (center) $\mathcal{C}$ and paths $\{P_i\}$ in PG from each $\mathcal{T}_i$ to $\mathcal{C}$, minimizing*

$$\text{cost}(\mathcal{C}, \mathfrak{T}) = \sum_{\mathcal{T}_i \in \mathfrak{T}} |P_i|$$

Ideally, we want $|P_i|$ to be $\mathrm{d}_{\text{pf}}(\mathcal{C}, \mathcal{T}_i)$. This raises the intermediate problem of computing the parallel flip distance of two triangulations. Computing the unit flip distance, where the goal is minimizing the number of unit flips, is NP-hard [8, 10]. The complexity of computing the parallel flip distance remains open. For the unit flip distance, exact methods [9] based on shortest paths and integer linear programs have been employed. To us, these approaches appeared ill-suited for large instances (although the winning team `Shadoks` [3] ultimately employed an optimal SAT solver with great success). Instead, we employ a greedy approach inspired by [5] for the sub-task of approximating $\mathrm{d}_{\text{pf}}(\cdot)$.

Based on this heuristic distance computation, we solve Problem 1 via a simulated-annealing-based approach. Our method is separated into three stages:

1. Compute an initial candidate center $\mathcal{C}$.
2. Improve $\mathcal{C}$ via simulated annealing, with parallel flip distance heuristics as a guide.
3. Use more expensive distance heuristics to improve $\text{cost}(\mathcal{C}, \mathfrak{T})$.

We describe our parallel flip distance heuristics in Section 2 and our simulated-annealing-based center search in Section 3. Finally, we present experimental results in Section 4.

2 Parallel Flip Distance Heuristics

We start with an observation on parallel flips. Given a triangulation $\mathcal{D}$, we define the *conflict graph* $G_{\mathcal{D}}$ as the graph whose vertices correspond to all flippable edges in $\mathcal{D}$, and in which two vertices are adjacent if the corresponding edges are part of the same triangle and hence cannot be part of the same parallel flip. Thus, parallel flips correspond to independent sets in $G_{\mathcal{D}}$. Based on this, the idea of our flip heuristics $H_u(\mathcal{D}_s, \mathcal{D}_t)$ can be summarized as follows:

1. Start with $\mathcal{D}_0 = \mathcal{D}_s$.
2. Compute a utility value $u(e)$ for each flippable edge $e \in \mathcal{D}_i$, such that flipping an edge with high utility is expected to produce a triangulation closer to the given target $\mathcal{D}_t$.
3. Find a maximum-weight independent set (MWIS) heuristically.
4. Apply the corresponding parallel flip operation to obtain a new triangulation $\mathcal{D}_{i+1}$.
5. Repeat Steps 2–4 with $\mathcal{D}_{i+1}$ until the target triangulation is reached.

Typically, the resulting flip distance estimate is asymmetric, i.e., $H_u(\mathcal{D}, \mathcal{D}') \neq H_u(\mathcal{D}', \mathcal{D})$.

Flip Utilities. Hanke et al. [5] observed that the number of proper crossings between edges of two triangulations is an upper bound on the (unit) flip distance. Consequently, they introduced a heuristic that aims to aggressively decrease the number of such crossings. This definition extends nicely to the parallel flip distance, and serves as the basis for our utility function: Let e be an edge connecting two vertices of $\mathcal{S}$, and let $\mathcal{D}_t$ denote the target triangulation. Define $\text{cr}_{\mathcal{D}_t}(e)$ as the number of proper crossings between e and the edges of $\mathcal{D}_t$. For a flippable edge e in $\mathcal{D}$, let $u_H(e, \mathcal{D}, \mathcal{D}_t) = \text{cr}_{\mathcal{D}_t}(e) - \text{cr}_{\mathcal{D}_t}(\text{flip}(e))$, while for non-flippable edges, let $u_H(e, \mathcal{D}, \mathcal{D}_t) = -\infty$. Note that if $\mathcal{D} \neq \mathcal{D}_t$, at least one edge has positive utility [5]. This heuristic is heavily geared towards the unit flip distance: it does not take into account that flipping an edge e blocks all other edges of its two triangles from being added to the current parallel flip. Consequently, we define a natural extension to this heuristic: Let

$$u_{2L}(e, \mathcal{D}, \mathcal{D}_t) = u_H(e, \mathcal{D}, \mathcal{D}_t) - \sum_{z \in \delta(e)} u_H(z, \mathcal{D}, \mathcal{D}_t),$$

where $\delta(e)$ denotes the (flippable) neighbors of e in the conflict graph $G_{\mathcal{D}}$. Our edge selection for the construction of a parallel flip is sequential, so during the selection process, some neighbors of an edge e may already be blocked by previously selected edges. Thus, we can dynamically exclude them from the penalty sum in $u_{2L}(e, \mathcal{D}, \mathcal{D}_t)$. The values cr, u_H, and u_{2L} naturally extend to multiple target triangulations by summing over all terminals.

MWIS Heuristics. Our main MWIS heuristic scans all flippable edges in decreasing order of utility $u(e, \mathcal{D}, \mathcal{D}_t)$ and adds an edge e if the solution remains an independent set and $u(e, \mathcal{D}, \mathcal{D}_t) > 0$. For more accuracy, at the expense of runtime, we refine the initial greedy solution via a local search that swaps selected edges with some of their non-selected neighbors, as long as it increases the total utility. This process ends when no beneficial swaps remain.

Randomization. We introduce randomization into the greedy edge selection algorithm and rerun the randomized parallel flip sequence computation multiple times to increase the overall accuracy. Concretely, given a parameter k (usually $k \leq 50$), we maintain a buffer of the k unblocked edges with the highest utility. At each selection step, one edge from the buffer is selected uniformly at random and added to the parallel flip before updating the buffer.

Engineering. The main runtime cost of the heuristic $H_u(\mathcal{D}_s, \mathcal{D}_t)$ is the initial computation of $\mathrm{cr}_{\mathcal{D}_t}(e)$ for edges $e \in \mathcal{D}_s$ with respect to the target $\mathcal{D}_t$. Here, a brute-force approach combined with an R-tree outperformed a Bentley-Ottmann sweep [2], which requires exact arithmetic for rational crossing points. For small instances ($|\mathcal{S}| \leq 1000$), we speed up computations of the form $H_u(\mathcal{C}, \mathcal{T})$, with current center $\mathcal{C}$ and terminal $\mathcal{T}$, by precomputing a table that contains the number of crossings $\mathrm{cr}_{\mathcal{T}}(e)$ for every possible edge $e \in \binom{\mathcal{S}}{2}$. As the size of this table is quadratic in $|\mathcal{S}|$, we do not compute it for larger instances.

3 Center Search via Simulated Annealing

To optimize the chosen center, we employ simulated annealing [6] with a fixed time horizon t_{total}. Simulated annealing is a local search technique that iteratively explores the solution space by moving to randomly generated neighboring solutions. Improving moves are always accepted, while worsening moves are accepted with a probability $P_{\text{accept}}(\Delta, t)$ that depends on the objective change Δ and the normalized progress $t = t_{\text{current}}/t_{\text{total}}$. This probability is controlled by a temperature parameter that evolves according to a cooling schedule.

Cooling Schedule. We use an exponential cooling schedule that quickly reduces the temperature such that, in the final quarter of the runtime, the algorithm effectively behaves as a greedy local search. Concretely, we set $T_{\text{quarter}} = -\frac{1}{\ln 0.02}$, and $T(t) = T_0 \cdot \left(\frac{T_{\text{quarter}}}{T_0}\right)^{t/0.75}$, ensuring $T(0.75) = T_{\text{quarter}}$. To dampen the impact of large cost increases, we use a sublinear transformation of Δ and accept moves with probability $P_{\text{accept}}(\Delta, t) = \exp\left(-\frac{\Delta^{2/3}}{T(t)}\right)$. With $T_0 = 30$, this yields strong exploration early on and greedy behavior in the final quarter.

Initial Centers. We use four different strategies to generate the initial center:

1. Delaunay: A simple choice for the initial center is the Delaunay triangulation.
2. Path: For each terminal pair $\mathcal{T}, \mathcal{T}' \in \mathfrak{T}$, compute a short heuristic flip path, evaluate all triangulations along the paths and select the best one as the initial center.
3. Intersection: For each edge $e \in \binom{\mathcal{S}}{2}$, count the total number $\mathrm{cr}_{\mathfrak{T}}(e)$ of crossings with all terminals. Construct the center greedily by adding edges in increasing order of $\mathrm{cr}_{\mathfrak{T}}(e)$.
4. Frequency: Each edge e in $\binom{\mathcal{S}}{2}$ is assigned a frequency based on how many terminals contain e. The initial triangulation then results from greedily selecting edges in decreasing order of frequency, using edges with frequency 0 if necessary.

Neighbor Generation. For small instances, where evaluating the flip heuristic is fast, we generate neighbors randomly: We pick a random number k, then we pick k flippable edges uniformly at random and apply the flips in random order, skipping blocked edges.

For larger instances, increased running time of the distance heuristic results in fewer candidates $\mathcal{C}$ being evaluated, so in later stages, we use an informed neighbor generation: We select a random subset $\mathfrak{T}' \subset \mathfrak{T}$, compute $u_H(e, \mathcal{C}, \mathfrak{T}')$ for all flippable edges and select a random greedy flip with a (small) k-buffer. The transition from uninformed to informed

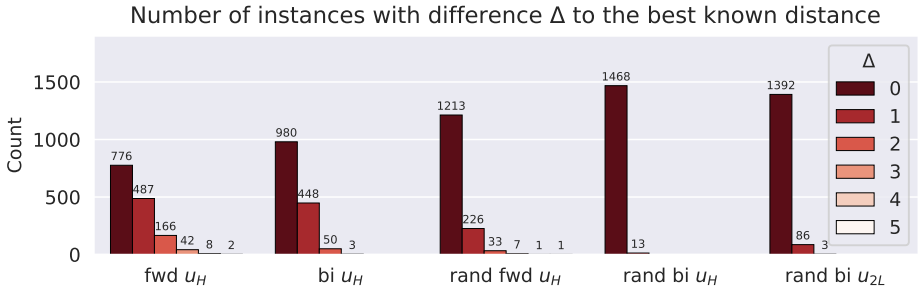


Figure 2 Comparison of deterministic and randomized heuristics: only-forward u_H, bidirectional u_H, only-forward randomized u_H, bidirectional randomized u_H, and bidirectional randomized u_{2L}.

neighbor generation is again controlled by a schedule that gradually increases the probability of selecting an informed neighbor. In the mostly greedy last quarter of the search, we substantially increase the buffer size k for informed flips, reintroducing randomness.

Heuristic Selection. We evaluated the simulated annealing algorithm with many heuristic combinations. The one we found to be the most promising was the following: First, during the exploration phase, we use u_H or u_{2L} to compute only the asymmetric deterministic variants of the heuristics, focusing on computational speed. Then, in the final (greedy) phase of the algorithm, we transition to the full ensemble of heuristics, including all introduced randomization schemes, and their opposites $H_u^{\text{opp}}(\mathcal{D}, \mathcal{D}')$., i.e., $H_u^{\text{opp}}(\mathcal{D}, \mathcal{D}') = H_u(\mathcal{D}', \mathcal{D})$.

4 Experiments

The competition data set consists of 250 instances, grouped into 100 `random`, 101 `woc`, and 49 `rirs` instances. The `random` and `woc` instances are small (15–320 points and 2–20 terminals), whereas the `rirs` instances are substantially larger (500–12 500 points and 20–200 terminals). All experiments were conducted on nodes of the HPC cluster Marvin (University of Bonn), each equipped with 1024 GB RAM and two Intel Xeon Sapphire Rapids 2.10 GHz CPUs.

Parallel Flip Distance Heuristics. To evaluate the quality of the heuristics, we generated up to ten distinct terminal pairs for each instance (1481 samples in total) and computed the parallel flip distance using u_H and u_{2L}, combining greedy and local search to compute the MWIS. The computed flip distances range from 1 to 37 and increase with the instance size.

We observed that the distance values of the heuristics using different utility functions differ in only 6% of instances. Hence, in Figure 2, we focus on u_H. As expected, applying the heuristic in a single direction (H_{u_H}) performs worst, with some outliers deviating by up to five flips from the best known values. Taking the minimum of both directions ($\min(H_{u_H}, H_{u_H}^{\text{opp}})$) increases the number of best values found by 25% and eliminates most of the extreme outliers.

Notably, even the unidirectional randomized variant outperforms the bidirectional deterministic one, yielding a further performance gain of 25%, although some outliers with large deviations from the best value remain. Minimizing over both directions yields the best solutions for most instances; using u_{2L} instead of u_H performs better in only thirteen cases.

Center Search with Simulated Annealing. We select one representative instance from each data set. Figure 3 shows the progression of the objective value during the center search for different configurations. All configurations use u_H for exploration and the full heuristic

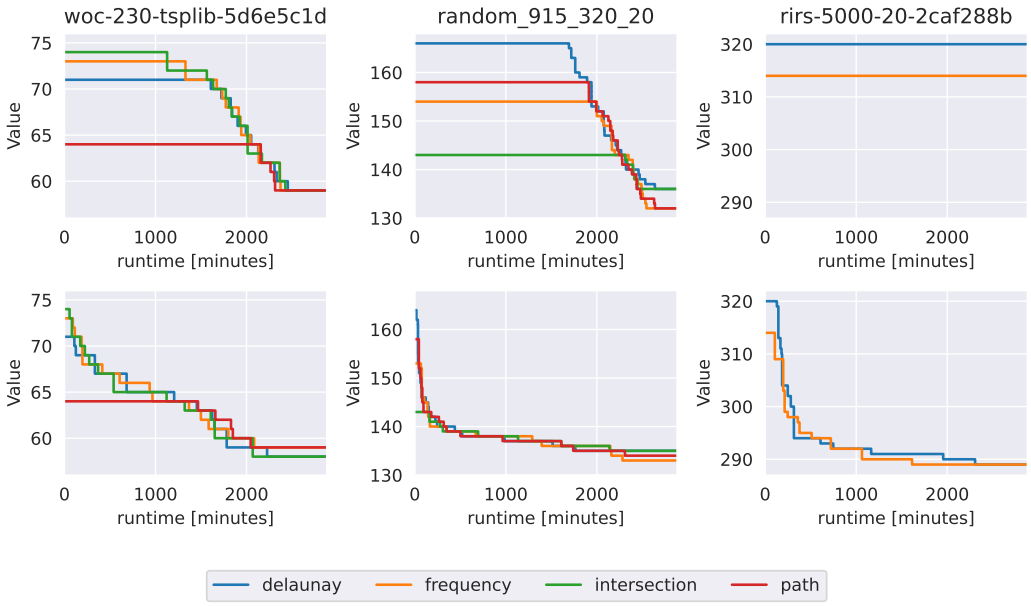


Figure 3 Objective value progression for different configurations during simulated annealing with varying initial centers. Top: uninformed neighbors; bottom: uninformed and informed neighbors.

ensemble in the final quarter. The main difference is in the neighbor selection: the upper row uses only random neighbors, while the bottom row incorporates informed neighbors. As expected, purely random neighbors do not improve the current best value during the exploration phase, but then improve the solution during the final (greedy) search. In contrast, the informed search shows gradual improvements, albeit with a higher risk of getting trapped in local optima. For small instances, the random configuration achieves slightly better final results, but unlike the informed search, it fails to find improvements on the `rirs` instance.

Figure 4 shows that the final objective value changes little with longer runtime for either neighbor selection and can even get worse due to randomization. Further experiments indicate that two to three days is a sweet spot, and that reruns should be favored over extended optimization time. Overall, our example instances are representative: on `random`, uninformed usually wins; on `rirs`, informed always wins; and on `woc`, both outcomes were observed.

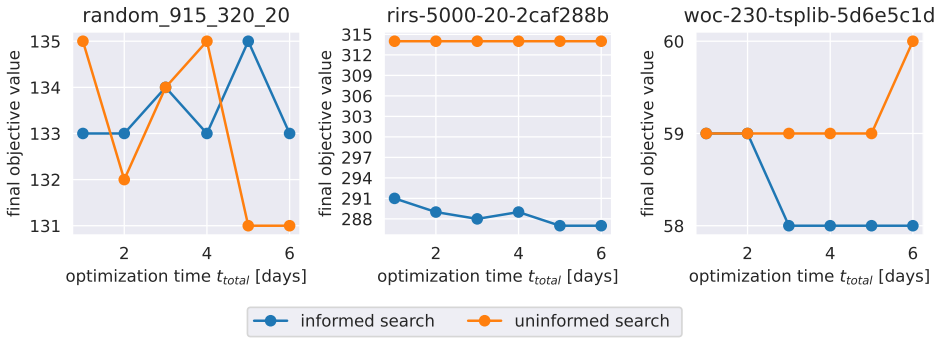


Figure 4 Final objective value after running for t_{total} days starting at the frequency center.

References

1 Oswin Aichholzer, Joseph Dorfer, Sándor P. Fekete, Phillip Keldenich, Peter Kramer, and Stefan Schirra. Central triangulation under parallel flip operations: The cg:shop challenge 2026, 2026. arXiv:2603.18812.

2 Jon L. Bentley and Thomas A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28(9):643–647, 1979. doi:10.1109/TC.1979.1675432.

3 Guilherme Dias da Fonseca and Yan Gerard. Shadoks approach to parallel reconfiguration of triangulations. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, 2026. doi:10.4230/LIPIcs.SoCG.2026.107.

4 Jérôme Galtier, Ferran Hurtado, Marc Noy, Stephane Perennes, and Jorge Urrutia. Simultaneous edge flipping in triangulations. *Int. J. Comput. Geom. Appl.*, 13(2):113–133, 2003. doi:10.1142/S0218195903001098.

5 Sabine Hanke, Thomas Ottmann, and Sven Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2:570–579, April 1996. doi:10.3217/JUCS-002-08-0570.

6 S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. doi:10.1126/science.220.4598.671.

7 Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365–372, 1972. doi:10.1016/0012-365X(72)90093-3.

8 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. 24th Canadian Conference on Computational Geometry (CCCG'12). doi:10.1016/j.comgeo.2014.11.001.

9 Philip Mayer and Petra Mutzel. Engineering A* Search for the Flip Distance of Plane Triangulations. In *22nd International Symposium on Experimental Algorithms (SEA 2024)*, volume 301 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SEA.2024.23.

10 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. doi:10.1016/j.comgeo.2014.01.001.

Shadoks Approach to Parallel Reconfiguration of Triangulations

Guilherme D. da Fonseca
LIS, Aix-Marseille Université, France

Fabien Feschet
LIMOS, Université Clermont Auvergne, France

Yan Gerard
LIMOS, Université Clermont Auvergne, France

Abstract

We describe the methods used by Team Shadoks to win the CG:SHOP 2026 Challenge on parallel reconfiguration of planar triangulations. Our approach combines exact methods based on SAT with several greedy heuristics, and also makes use of SAT and MaxSAT for solution improvement.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Exact algorithm, SAT, MaxSAT, heuristic, computational geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.107

Category CG Challenge

Related Version *Full Version*: `https://arxiv.org/abs/2603.21293`

Supplementary Material *Software (Source Code)*: `https://github.com/gfonsecabr/shadoks-CGSHOP2026`, archived at `swh:1:dir:14b1868ef49dcd479ff783d738ee448d33be9221`

Acknowledgements We would like to thank the Challenge organizers and other competitors for their time, feedback, and making this whole event possible. We would like to thank Hélène Toussaint, Raphaël Amato, Boris Lonjon, and William Guyot-Lénat from LIMOS, as well as the Qarma and TALEP teams and Manuel Bertrand from LIS, who continue to make the computational resources of the LIMOS and LIS clusters available to our research. We would also like to thank Aldo Gonzalez-Lorenzo for the very useful discussion on SAT models.

1 Introduction

The CG:SHOP Challenge is an annual competition in geometric optimization. In 2026, the challenge focuses on a reconfiguration problem between planar triangulations. Our team, called *Shadoks*, won first place with the best solution (among the 28 participating teams) to 249 instances out of 250 instances and provably optimal solutions to 189 instances.

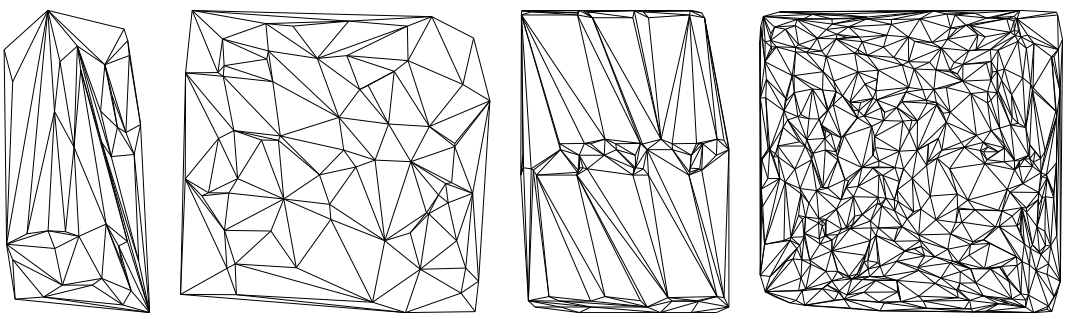


Figure 1 Our best centers to instances `random_78_40_10`, `woc-70-random-9a7d18d3`, `woc-90-tsplib`, and `rirs-500-50-23d00ec5`, respectively.

© Guilherme D. da Fonseca, Fabien Feschet, and Yan Gerard;
licensed under Creative Commons License CC-BY 4.0

In this paper, we outline the exact methods that we employed. The heuristic solvers are only presented in the full version due to space limitations. We start with some definitions that allow us to describe the problem. Throughout, we consider triangulations of a common point set $S \subset \mathbb{R}^2$.

Given a triangulation T, a *unit flip* is the operation that removes an edge $e \in T$ and adds an edge e'. The unit flip of the edge $e = uv$ considers the empty convex quadrilateral u, u_2, v, v_2 and replaces its diagonal uv it by the other diagonal u_2v_2 (Figure 2).

Similarly, a *parallel flip* removes a set of edges $E \subset T$ and adds a set of edges E', in a way that $T' = T \setminus E \cup E'$ is a triangulation, with the condition that no two edges of E are in the same triangle in T. A *path* of *length* ℓ is a sequence of triangulations $T_0, \ldots, T_\ell$ such that for all i, the triangulation T_{i+1} is obtained from T_i by performing a parallel flip.

An *instance* is a set $S \subset \mathbb{R}^2$ of n points and a set $\mathcal{T} = \mathbf{T}_1, \ldots, \mathbf{T}_{|\mathcal{T}|}$ of triangulations of S, called *input triangulations*. A *solution* is a set of paths $P_1, \ldots, P_{|\mathcal{T}|}$ such that P_i starts at $\mathbf{T}_i$ for all i and all paths end in a common triangulation called *center*. The goal is to find a solution that minimizes the *objective value* defined as the sum of the lengths of its paths.

During the competition, the organizers provided a total of 250 instances, with n ranging from 15 to 12,500 points and $|\mathcal{T}|$ ranging from 2 to 200 triangulations. The 250 instances are divided into three classes: 100 `random` instances, 101 `woc` instances, and 49 `rirs` instances. The former two instances have up to 320 points and 2 to 20 input triangulations (hence, we call them `small` instances), while the latter have 500 to 12500 points and 20 to 200 input triangulations. The centers of some of our best solutions are presented in Figure 1. Additional details about the challenge can be found in the organizers' survey paper [2].

Our best solvers heavily rely on the SAT solver `CaDiCal` [5] and the MaxSAT solver `EvalMaxSAT` [3]. Nevertheless, we also developed heuristics that do not rely on any external solver, which are important to find initial solutions to large instances, which are then improved by roughly 10% using SAT and MaxSAT solvers., which are presented in the full version.

Other team strategies include SAT solvers for small instances [4], heuristics [9], and local search and simulated annealing [7] to improve solutions.

2 Exact Algorithms

2.1 Path SAT Formulation

Next, we describe a SAT formulation for the following decision problem. The input is a set S of n points, an integer ℓ and two triangulations T_0, T_ℓ. The output is whether there exists a path $T_0, \ldots, T_\ell$ of length ℓ.

We define two types of variables. For $i = 0, \ldots, \ell$ and for $u \neq v \in S$, we define an *edge variable* $e(u, v, i)$. The variable $e(u, v, i)$ represents that the edge uv is in the triangulation T_i. There are $O(n^2\ell)$ edge variables. It would be possible to define a SAT formulation using only such variables. However, a SAT formulation that performed much better in our experiments uses a second type of variable.

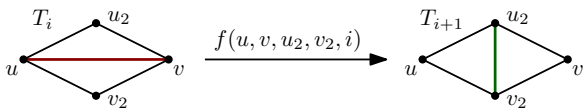


Figure 2 Illustration of a flip and the associated variable $f(u, v, u_2, v_2, i)$.

We say that a convex quadrilateral is *empty* if it contains no point of S except for its vertices. For $i = 0, \ldots, \ell$-1 and for an empty convex quadrilateral u, u_2, v, v_2, we introduce *flip variables* $f(u, v, u_2, v_2, i)$, which is true if and only if uv is in triangulation T_i and u_2v_2 is in triangulation T_{i+1}, as shown in Figure 2. Notice that if the points are uniformly distributed, then the number of empty convex quadrilaterals is $\Theta(n^2)$ [8], which means that for uniformly distributed points, the number of flip variables is also $O(n^2\ell)$. However, the number of flip variables is $\Theta(n^4\ell)$ if the points are in convex position (which is not the case for the challenge instances). Next, we describe the different types of clauses.

Start and target. For every edge variable $e(u, v, 0)$, we have the clause $e(u, v, 0)$ if $uv \in T_0$ and $\neg e(u, v, 0)$ if $uv \notin T_0$. For every edge variable $e(u, v, \ell)$, we have the clause $e(u, v, \ell)$ if $uv \in T_\ell$ and $\neg e(u, v, \ell)$ if $uv \notin T_\ell$.

Flips need edges. For every flip variable $f(u, v, u_2, v_2, i)$, we have the 5 binary CNF clauses translating the implication

$$f(u, v, u_2, v_2, i) \implies e(u, v, i) \wedge e(u, v_2, i) \wedge e(u, u_2, i) \wedge e(v, v_2, i) \wedge e(v, u_2, i).$$

Flips keep edges. Similarly, or every flip variable $f(u, v, u_2, v_2, i)$, we have the 5 binary CNF clauses translating the implication

$$f(u, v, u_2, v_2, i) \implies e(u_2, v_2, i{+}1) \wedge e(u, v_2, i{+}1) \wedge e(u, u_2, i{+}1) \wedge e(v, v_2, i{+}1) \wedge e(v, u_2, i{+}1).$$

Flips flip edges. For every flip variable $f(u, v, u_2, v_2, i)$, we have the binary clauses translating

$$f(u, v, u_2, v_2, i) \implies \neg e(u, v, i + 1).$$

Edge changes require flips. The last type of clause is the only one that has more than 2 variables in CNF form. It states that if the edge variable changes from triangulation i to $i + 1$, then there must be a flip. The $\bigvee$ below considers all values of the subscript that define existing flip variables. We have two such clauses for each edge variable:

$$e(u, v, i) \wedge \neg e(u, v, i + 1) \implies \bigvee_{u_2, v_2} f(u, v, u_2, v_2, i) \text{ and}$$

$$\neg e(u_2, v_2, i) \wedge e(u_2, v_2, i + 1) \implies \bigvee_{u, v} f(u, v, u_2, v_2, i).$$

Eliminating variables and clauses. The number of variables and clauses grows very fast, even though the number of clauses is linear in the number of variables. Next, we show how to eliminate many variables from the model. All eliminated variables are defined as *false* and the clauses that become tautologies are eliminated. If a CNF clause becomes empty, then the problem is unsatisfiable.

The following theorem is easy to prove and implies that $\Omega(\log n)$ parallel flips are sometimes necessary to reconfigure two triangulations of n points, even when the points are in convex position. We say that two segments *cross* if they intersect at a point that is not an endpoint of either segment.

▶ **Theorem 1.** *Consider two triangulations T, T' of S such that a parallel flip transforms T into T' and a segment s with endpoints in S. Let χ, χ' respectively denote the number of edges of T, T' crossed by s. We then have $\chi' \geq \lfloor \chi/2 \rfloor$.*

Consequently, we only define the variable $e(u, v, i)$ when uv crosses strictly less than 2^i edges of T_0 and strictly less than $2^{\ell-i}$ target edges. We only define flip variables when a certain set of edge variables is defined: $f(u, v, u_2, v_2, i)$ is only defined when $uv, uv_2, uu_2, vv_2, vu_2$, are all defined at i and $u_2v_2, uv_2, uu_2, vv_2, vu_2$, are all defined at $i + 1$.

2.2 Solution SAT Formulation

Next, we describe a SAT formulation for the following decision problem. Recall that an instance is a set S of n points and a list $\mathcal{T}$ of input triangulations $\mathbf{T}_1, \ldots, \mathbf{T}_{|\mathcal{T}|}$. The input of the decision problem is an instance and $|\mathcal{T}|$ integers $\ell_1, \ldots, \ell_{|\mathcal{T}|}$. The output is whether there exists a solution $P_1, \ldots, P_{|\mathcal{T}|}$ such that path P_i has length ℓ_i for all i.

We model the $|\mathcal{T}|$ paths $P_1, \ldots, P_{|\mathcal{T}|}$ independently as before, starting path P_i at the input triangulation $\mathbf{T}_i$. The final triangulation of each path is unknown, but the same edge variables are used for the final triangulation of every path, since a valid solution requires that all paths end in the same triangulation. It is easy to see that the SAT formulation is satisfiable if and only if there exists a solution with the given lengths.

2.3 Lower Bound

In order to obtain an exact solution to an instance $\mathcal{I}$, we start by computing a lower bound to its objective value. We say that the *distance* between two triangulations T, T' is the length of the shortest path from T to T'. We create a complete directed graph $G(\mathcal{I})$ with edge lengths as follows. The vertices are the triangulations $\mathcal{T}$ and the length of each edge is the distance between the corresponding triangulations. A *cycle packing* of G is a collection of vertex-disjoint directed cycles, i.e. a subset of edges such that each vertex has at most one outgoing and at most one incoming edge in the subset. The graph is directed to allow for cycles with only 2 edges. The *length of a cycle* is the sum of the lengths of its edges, and the *length of a cycle packing* is the sum of the lengths of its cycles. The maximum length cycle packing can be solved in polynomial time using a reduction to maximum weight bipartite matching [6]. The following theorem is easy to show.

▶ **Theorem 2.** *Given an instance $\mathcal{I}$, the objective value of a solution is at least the length of any cycle packing of $G(\mathcal{I})$ divided by 2.*

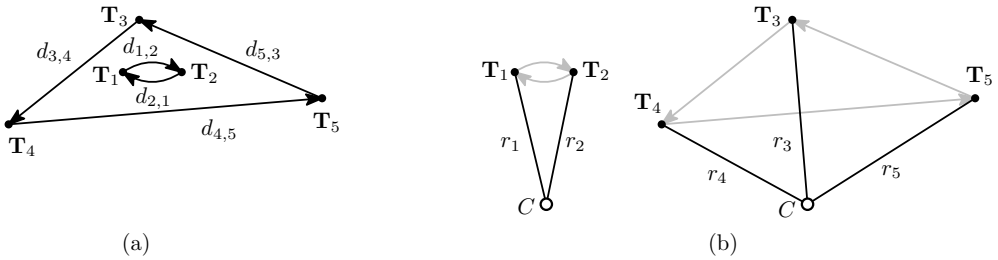


Figure 3 (a) A cycle packing. (b) Illustration of the proof. In this example,$d_{1,2} \leq r_1 + r_2$, $d_{2,1} \leq r_2 + r_1$, $d_{3,4} \leq r_3 + r_4$, $d_{4,5} \leq r_4 + r_5$, and $d_{5,3} \leq r_5 + r_3$ by triangle inequality.

2.4 The Exact Solver

First, we use the exact path formulation from Section 2.1 to calculate the distance between all $\binom{\mathcal{T}}{2}$ pairs of input triangulations using a SAT solver (in our case, `CaDiCal` [5]). It is easy to formulate the problem of finding a maximum length cycle packing as a weighted MaxSAT problem, which provides a lower bound b to the objective value (see Section 2.3). We solve this problem using a weighted MaxSAT solver (in our case `EvalMaxSAT` [3]). We then use backtracking to list all integer solutions to $\ell_0, \ldots, \ell_{|\mathcal{T}|} = b$ that satisfy $\ell_i + \ell_j \geq \text{distance}(T_i, T_j)$. We use the SAT formulation from Section 2.2 to test the existence of a solution with the given lengths $\ell_0, \ldots, \ell_{|\mathcal{T}|}$, again using the `CaDiCal` SAT solver. If a solution is found, then it is optimal. Otherwise, we increment b and repeat. Notice that b is always a lower bound to the objective value. Hence, if a solution obtained by a heuristic attains this lower bound, then it is optimal.

2.5 Happy Edges Conjecture

The happy edges conjecture [1] is a general conjecture that is *false* for some reconfiguration problems and *true* for others.

▶ **Conjecture 3.** *For any pair of configurations T, T', there exists a shortest path between T, T' where the edges that are common to both T and T' appear in all intermediate configurations.*

The conjecture is *false* for triangulations under unit flips and arbitrary points [10] but *true* when the points are in convex position [11]. Our experiments lead us to believe that the conjecture is *true* for parallel flips, as we could not find a counterexample.

When computing a path of length ℓ from T_0 to T_ℓ using SAT, for every edge uv that appears in both T_0 and T_ℓ, we add clauses $e(u, v, i)$ that force the variable to be true for all i. More importantly, we then eliminate every edge variables corresponding to edges that cross uv. The same idea can be applied to the SAT formulation that finds a solution, but then only the edges that appear in all input triangulations are forced to be true for all i, and again the edges that cross them are eliminated.

Furthermore, when computing a path, for every edge $uv \in T_\ell$, we eliminate flip variables that remove uv, i.e. $f(u, v, u_2, v_2, i)$ for all u_2, v_2, i. Similarly, for every edge $u_2v_2 \in T_0$, we eliminate flip variables that insert u_2v_2, that is $f(u, v, u_2, v_2, i)$ for all u, v, i.

3 Results

In this section, we present the computational results that we obtained with our implementation of the aforementioned algorithm. In Section 3.1, we present the results on computing short paths between two given triangulations. In Section 3.2, we present our exact solver, with and without the happy edges conjecture.

The solvers were coded in C++ and compiled with GCC and run a single thread. During the competition, they were executed on several Linux computers, either using GNU Parallel [12] for local executions or Slurm [13] for cluster executions. It was very useful to have access to machines with 128GB or more RAM to solve large SAT formulations with `CaDiCal` [5], which has been able to solve SAT instances with more than 50 million variables and 500 million clauses. The time measurements on this paper use an AMD Ryzen 9 9900X CPU and ASUS TUF B650M motherboard with 128GB of RAM running Fedora Core 43.

Table 1 Length and computation time (in milliseconds) for a path from the Delaunay triangulation to the first input triangulation of the `rirs-`n`--20` instance for different values of the number of points n. The Heuristic column shows the smallest length obtained by applying four heuristics (greedy (forward/backward) and squeaky-wheel (forward/backward)) described in the full version of the paper and the total time to run the four heuristics. The SAT columns correspond to SAT formulation with or without the happy edges conjecture, unless it takes too long.

	Heuristic		SAT happy		SAT exact	
n	length	total time	length	time	length	time
500	**12**	70.9	**12**	1512	**12**	10191
1000	12	77.6	**11**	6806	**11**	78264
1500	12	100	**11**	19302	**11**	201353
2000	**13**	208	**13**	16937	**13**	485177
3000	**14**	695	**14**	140729	·	·
4000	**15**	1340	**15**	101402	·	·
5000	**16**	656	**16**	1037840	·	·
6000	**16**	1891	**16**	271081	·	·
7000	17	777	**16**	471389	·	·

3.1 Path Calculation

Computing short paths between two given triangulations is a key component to obtain good solutions. Typically, these paths are computed with an input triangulation as one extreme, and a triangulation that makes a reasonably good center as the other extreme. Table 1 shows the best length of the path obtained with our heuristics (described in the full version of the paper) with the associated running times compared to our SAT formulation with and without the happy edges conjecture. The paths are computed from the Delaunay triangulation that makes a reasonably good (but not very good) center to the first input triangulation of several instances. The SAT paths are obtained by first running a heuristic in both directions, and then iteratively decreasing the path length with our SAT solution.

3.2 Exact Solutions

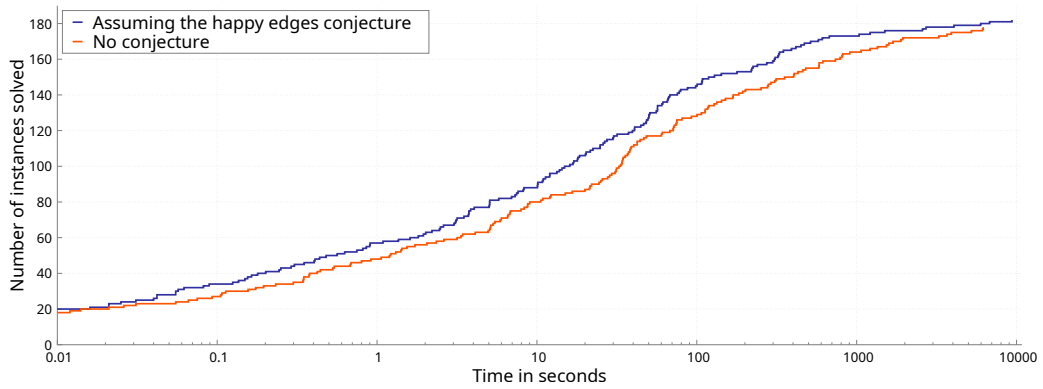


Figure 4 Number of exact solutions found as a function of the running time over 3 hours of execution with and without assuming the happy edges conjecture.

We end the paper with the performance of our exact solver. Figure 4 plots the number of exact solutions found as a function of runtime, with and without the happy edges conjecture. Since the solution values are identical in both cases, the happy edge conjecture holds.

References

1. Oswin Aichholzer, Brad Ballinger, Therese Biedl, Mirela Damian, Erik D Demaine, Matias Korman, Anna Lubiw, Jayson Lynch, Josef Tkadlec, and Yushi Uno. Reconfiguration of non-crossing spanning trees, 2022. doi:10.48550/arXiv.2206.03879.
2. Oswin Aichholzer, Joseph Dorfer, Sándor P. Fekete, Phillip Keldenich, Peter Kramer, and Stefan Schirra. Central triangulation under parallel flip operations: The CG:SHOP challenge 2026, 2026. arXiv:2603.18812.
3. Florent Avellaneda. A short description of the solver EvalMaxSAT. *MaxSAT Evaluation*, 8:364, 2020.
4. Lorenzo Battini and Marko Milenković. ETH flippers approach to parallel reconfiguration of triangulations: SAT formulation and heuristics. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, 2026. doi:10.4230/LIPIcs.SoCG.2026.105.
5. Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL 2.0. In *Computer Aided Verification - 36th International Conference, CAV*, volume 14681 of *LNCS*, pages 133–152, 2024. doi:10.1007/978-3-031-65627-9_7.
6. P Biró, D Manlove, and R Rizzi. Maximum weight cycle packing in optimal kidney exchange programs, university of glasgow, department of computing science. Technical report, Technical Report TR–2009–298, 2009.
7. Jacobus Conradi, Benedikt Kolbe, Philip Mayer, Jonas Sauer, and Jack Spalding-Jamieson. Engineering greedy heuristics and simulated annealing methods for the median triangulation under the parallel flip distance. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, 2026. doi:10.4230/LIPIcs.SoCG.2026.106.
8. Ruy Fabila-Monroy, Clemens Huemer, and Dieter Mitsche. The number of empty four-gons in random point sets. *Electronic Notes in Discrete Mathematics*, 46:161–168, 2014. doi:10.1016/j.endm.2014.08.022.
9. Jaegun Lee, Seokyun Kang, Hyeonseok Lee, Hyeyun Yang, and Taehoon Ahn. CG#Hunters approach to central triangulation under parallel flip operations. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, 2026. doi:10.4230/LIPIcs.SoCG.2026.108.
10. Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. doi:10.1016/j.comgeo.2014.01.001.
11. Daniel D. Sleator and William P. Tarjan, Robert E.and Thurston. Rotation distance, triangulations, and hyperbolic geometry. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 122–135, 1986.
12. O. Tange. GNU parallel – The command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47, February 2011. URL: http://www.gnu.org/s/parallel.
13. Andy B Yoo, Morris A Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60, 2003. doi:10.1007/10968987_3.

CG#Hunters Approach to Central Triangulation Under Parallel Flip Operations

Jaegun Lee
Department of Computer Science and Engineering, Pohang University of Science and Technology, Republic of Korea

Seokyun Kang
Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Republic of Korea

Hyeonseok Lee
Department of Computer Science and Engineering, Ulsan National Institute of Science and Technology, Republic of Korea

Hyeyun Yang
School of Computational Sciences, Korea Institute for Advanced Study, Seoul, Republic of Korea

Taehoon Ahn[1]
Department of Computer Science, Sookmyung Women's University, Seoul, Republic of Korea

Abstract

In the CG:SHOP 2026 Challenge, the goal is to compute a central triangulation for a given set of triangulations on the same point set while minimizing the sum of parallel flip distances. To address the problem, our team (CG#Hunters) constructs an initial solution by iteratively applying parallel flips to reduce the total number of crossings between the triangulations until none remain. To optimize these solutions, we shorten the paths by using a score-based greedy edge selection and refine the central triangulation via a large scale neighborhood search. Additionally, a representative-set-based approach is utilized to efficiently handle large instances. With these combined approaches, we achieved third place by successfully computing central triangulations with sufficiently short parallel flip paths for all 250 instances.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Central triangulation, Parallel flip operations, Crossing number, Large scale neighborhood search, Representative set

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.108

Category CG Challenge

Supplementary Material *Software (Source Code)*: `https://github.com/jagunlee/CGSHOP26.git` archived at `swh:1:dir:83f9595d8b77d04fc9756fda5f9eb46f0521604d`

Funding *Jaegun Lee, Seokyun Kang*: Partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (RS-2023-00219980), and the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II191906, Artificial Intelligence Graduate School Program (POSTECH)) and (No. 2017-0-00905, Software Star Lab (Optimal Data Structure and Algorithmic Applications in Dynamic Geometric Environment)).
Hyeonseok Lee: Supported by the Korea institute of Human Resources Development in Science and Technology (KIRD) grant funded by the Korea government (MSIT) (No. RS-2025-02215586).
Hyeyun Yang: Supported by a KIAS Individual Grant (CG102001) at Korea Institute for Advanced Study.

[1] Corresponding author

© Jaegun Lee, Seokyun Kang, Hyeonseok Lee, Hyeyun Yang, and Taehoon Ahn;
licensed under Creative Commons License CC-BY 4.0

Acknowledgements We would like to thank the challenge organizers and fellow competitors for their efforts. We are also grateful to Hee-Kap Ahn and Antoine Vigneron for providing computational resources at POSTECH and UNIST, and to the Center for Advanced Computation at Korea Institute for Advanced Study for help computing resources.

1 Introduction

An edge flip of a triangulation T is an operation that removes a shared edge e between two adjacent triangles forming a convex quadrilateral, and replaces it with the opposite diagonal of the quadrilateral. Given two triangulations T_1, T_2 on the same point set P, there exists a flip path from T_1 to T_2, which is a sequence of flips applied to T_1 to make T_2. There has been a fair amount of work on computing flip paths between two triangulations [2, 3, 5, 6], and it is known to be NP-complete to compute a shortest one [7].

Taking a step further, the CG:SHOP 2026 Challenge considers a *parallel flip* operation, which is a set of multiple edge flips that can be performed simultaneously because they do not share any triangles. We call a sequence of parallel flips a *parallel flip path*. In the problem, we are given a set of triangulations on the same point set and the goal is to compute a central triangulation on the same point set while minimizing the sum of the lengths of parallel flip paths from each triangulation. The problem is formally defined as follows.

▶ **Problem 1.** *Given a problem instance $I = (P, \mathcal{T})$, where P is a set of points in $\mathbb{R}^2$ and $\mathcal{T}$ is a set of triangulations of P, find a central triangulation C of P and a parallel flip path $\pi(T, C)$ from each $T \in \mathcal{T}$ to C while minimizing $\sum_{T\in\mathcal{T}} |\pi(T, C)|$.*

There were 250 instances of three types (*random-*, *woc-*, and *rirs-*) given. For more details on the instances, see the survey paper [1].

Preliminaries. We say that an edge e in a triangulation is *flippable* if it is shared by two triangles forming a convex quadrilateral. Two flippable edges of the same triangulation are *independent* if they do not share any triangle. Hence, a set of independent flippable edges can be chosen as a single parallel flip operation.

Our main approach relies on *the number of crossings* among a set of triangulations. We say that two edges are *crossing* if they intersect at a point in the interior of an edge. For an edge e and a triangulation T, we use $\mathrm{cr}(e, T)$ to denote the number of edges crossing e in T. Note that $\mathrm{cr}(e, T) = 0$ if $e \in T$. We slightly abuse the notation by letting $\mathrm{cr}(T, T') = \sum_{e\in T} \mathrm{cr}(e, T')$ denote the number of crossings between two triangulations T and T', and $\mathrm{cr}(\mathcal{T}) = \frac{1}{2} \cdot \sum_{T,T'\in\mathcal{T},T\neq T'} \mathrm{cr}(T, T')$ denote the total number of crossings among a set of triangulations $\mathcal{T}$.

2 Initial algorithm

First, we introduce our initial algorithm for computing an initial solution for each instance. Given an instance $I = (P, \mathcal{T})$, our algorithm runs while maintaining a (multi)-set of triangulations $\widehat{\mathcal{T}}$, starting with $\widehat{\mathcal{T}} = \mathcal{T}$. At each step, it selects one triangulation in $\mathcal{T}$ and applies a parallel flip to reduce the total number of intersections among the triangulations $\mathrm{cr}(\widehat{\mathcal{T}})$. The algorithm terminates when $\mathrm{cr}(\widehat{\mathcal{T}})$ becomes zero, which implies that the triangulations in $\widehat{\mathcal{T}}$ are the same. Then the algorithm returns the triangulation as the central triangulation C.

In doing so, we first use the following observation from [3].

▶ **Lemma 2** (Lemma 2 and 3 of [3]). *Let T, T' be two different triangulations on the same points. There exists an edge flip from T that reduces* $\mathrm{cr}(T, T')$.

Lemma 2 implies that we can always find a parallel flip from a triangulation $T \in \widehat{\mathcal{T}}$ that reduces the total number of intersections when $|\widehat{\mathcal{T}}| = 2$. To use this observation for an instance with $|\mathcal{T}| > 2$, we claim the following stronger argument.

▶ **Conjecture 3.** *Let $\widehat{\mathcal{T}}$ be a set of triangulations on the same point set. Then there exists a triangulation $T \in \widehat{\mathcal{T}}$ and an edge flip in T that reduces* $\mathrm{cr}(\widehat{\mathcal{T}})$, *unless* $\mathrm{cr}(\widehat{\mathcal{T}}) = 0$.

We presume that Conjecture 3 holds to design our algorithm, and so far during the contest, we have not encountered any counterexample. Using Conjecture 3, a single step of our algorithm works as follows. For each flippable edge e in each triangulation $T \in \widehat{\mathcal{T}}$, let e' be the edge obtained by flipping e in T. Then we define a *score* for e as the change in $\mathrm{cr}(\widehat{\mathcal{T}})$ after flipping e, that is, $\sum_{T' \in \widehat{\mathcal{T}} \setminus \{T\}} (\mathrm{cr}(e, T') - \mathrm{cr}(e', T'))$ and compute it. Observe that there is at least one positive-scored edge if Conjecture 3 holds.

After computing the scores of the edges, we compute a possible parallel flip to reduce $\mathrm{cr}(\widehat{\mathcal{T}})$ for each $T \in \widehat{\mathcal{T}}$. Within T, the edges to flip are chosen in a greedy manner. First, our algorithm sorts the positive-scored edges in T in descending order of their scores. Then, it scans through the sorted edges and picks an edge if it is independent of the already picked edges. After scanning, the algorithm chooses a triangulation with a possible parallel flip that reduces $\mathrm{cr}(\widehat{\mathcal{T}})$ the most and applies the parallel flip. Then the algorithm proceeds to the next step. See Figure 1.

By tracking the parallel flips applied to the triangulations during the algorithm, we can also obtain a parallel flip path $\pi(T, C)$ for each $T \in \mathcal{T}$. We mainly used this algorithm for obtaining initial solutions. However, this approach requires high computational cost. To address this, we also used a representative-set-based approach, which is described in Section 4.

3 Optimization

In this section, we address our two optimization approaches to improve our solutions: recomputing the paths and finding a new central triangulation.

3.1 Parallel flip path between two triangulations

In order to optimize the objective value $\sum_{T \in \mathcal{T}} |\pi(T, C)|$ for all triangulations in the given instance $I(P, \mathcal{T})$, one effective way is to compute shorter parallel flip paths. Our first optimization technique is specifically designed to compute a shorter parallel flip path from a given triangulation T to the central triangulation C without changing C itself.

To do so, we develop a core algorithm for computing a parallel flip path between two different triangulations T_1 and T_2 on the same point set. At each step, for each flippable edge in T_1 that is not present in T_2, a certain *score* of the edge with respect to T_2 is computed. These candidate edges are sorted in decreasing order of their scores. From this sorted list, a set of independent positive-scored edges is selected in a greedy manner. Finally, the edges in the set are flipped simultaneously, and the set of edges is recorded as an element for the sequence of the parallel flip path $\pi(T_i, T_j)$. The procedure above is repeated until $T_1 = T_2$.

During the competition, we used three different score measures to evaluate the geometric benefit of flipping an edge. For a flippable edge e in T_1 but not in T_2, let e' be the new edge obtained by flipping e. The measures for an edge e are defined as follows:

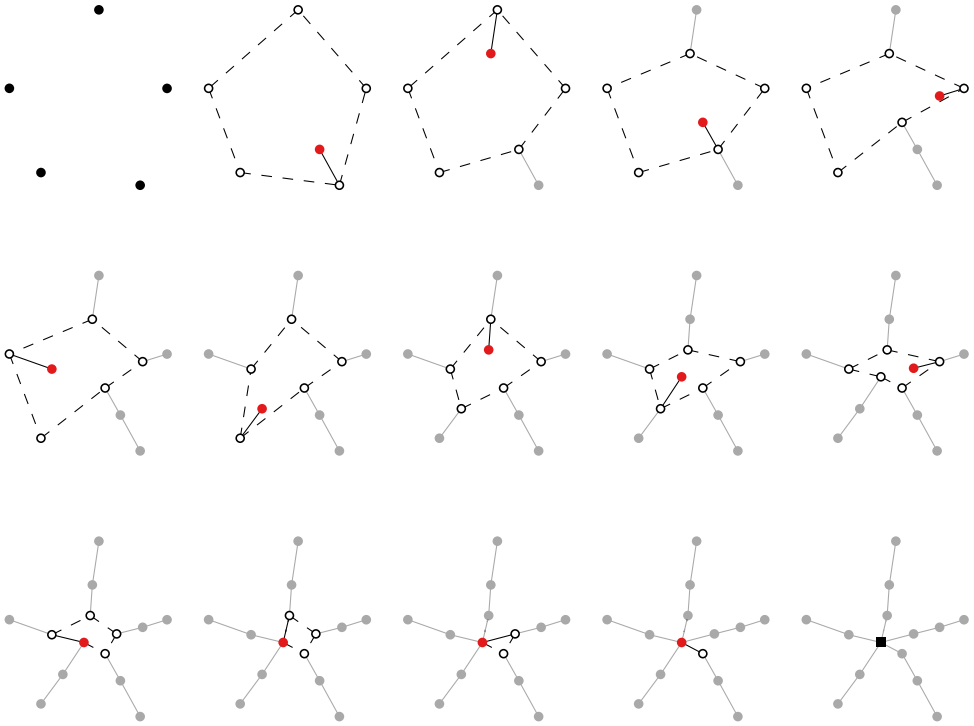


Figure 1 A visualization of the initial algorithm. Given an instance with five triangulations (black disks), the algorithm proceeds with maintaining $\widehat{\mathcal{T}}$ (empty disks). For each step, it picks one triangulation among $\widehat{\mathcal{T}}$ and applies a parallel flip (red disk) to the triangulation to reduce $\mathrm{cr}(\widehat{\mathcal{T}})$. The algorithm stops when triangulations in $\widehat{\mathcal{T}}$ are the same, and the triangulation (black box) is reported as a central triangulation.

1. $\mathrm{cr}(e, T_2)$: The absolute crossing number. This measure focuses on edges that have a high number of intersections with the target triangulation.
2. $\mathrm{cr}(e, T_2) - \mathrm{cr}(e', T_2)$: The absolute reduction in crossings. This evaluates the direct decrease in the number of intersections, favoring flips that consistently guide the current triangulation toward the target.
3. $(\mathrm{cr}(e, T_2) - \mathrm{cr}(e', T_2))/\mathrm{cr}(e, T_2)$: The relative reduction rate. This approach normalizes the reduction by the initial crossing count, favoring flips that resolve a larger proportion of their existing intersections.

Note that without proper cycle prevention, the algorithm may repeatedly select two edges alternatively, entering an infinite loop. See Figure 2. We were able to prevent it by prohibiting the edges flipped in the immediate previous step from being flipped again.

To derive the minimum parallel flip path $\pi(T_1, T_2)$, the aforementioned algorithm is extended by exploring diverse path generation strategies. For any given pair of triangulations, multiple candidate paths are generated using the following variations:

- **Forward/Reverse search:** Paths are computed not only from T_1 to T_2 but also from T_2 to T_1, with the latter being subsequently reversed.
- **Two-way search:** Paths are explored simultaneously from both triangulations. At each step, a parallel flip is greedily executed on the side yielding the higher score gain, allowing the two sequences to meet efficiently in the middle.
- **Local refinement:** After the candidate paths are generated, intermediate sub-paths between local states are iteratively extracted and recomputed using the strategies above.

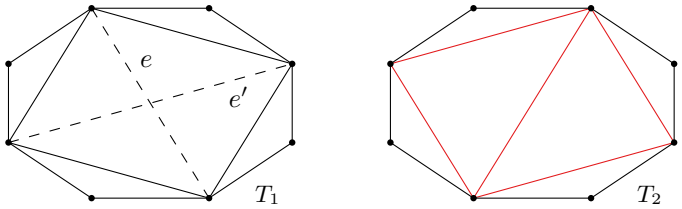


Figure 2 Two different triangulations T_1, T_2 for given a point set. Two dashed lines e and e' having $\mathrm{cr}(e, T_2) = 3$ and $\mathrm{cr}(e', T_2) = 3$ can be selected alternately in T_1 and fall into an infinite loop. Flipping e' right after flipping e is prevented allowing to flip other edges with crossing number 3.

3.2 Large scale neighborhood search

We optimize our solutions by finding a new central triangulation using large scale neighborhood search. Consider an instance $I = (P, \mathcal{T})$ and assume that we have already computed an initial solution consisting of a central triangulation C of I and parallel flip paths $\pi(T, C)$ for each $T \in \mathcal{T}$. Then for each $T \in \mathcal{T}$, a triangulation M_T along $\pi(T, C)$ is selected at random. Now, the initial algorithm is applied to compute a central triangulation C' for I', which is a new hypothetic instance $I' = (P, \mathcal{T}')$ where $\mathcal{T}' = \{M_T \mid T \in \mathcal{T}\}$ and to compute parallel flip paths $\pi'(M_T, C')$ for every $T \in \mathcal{T}$. A new solution for I is obtained by setting C' as a new central triangulation of I. The path from T to C' is then updated by concatenating $\pi(T, M_T)$−a sub-path in $\pi(T, C)$−with the new parallel flip path $\pi'(M_T, C')$.

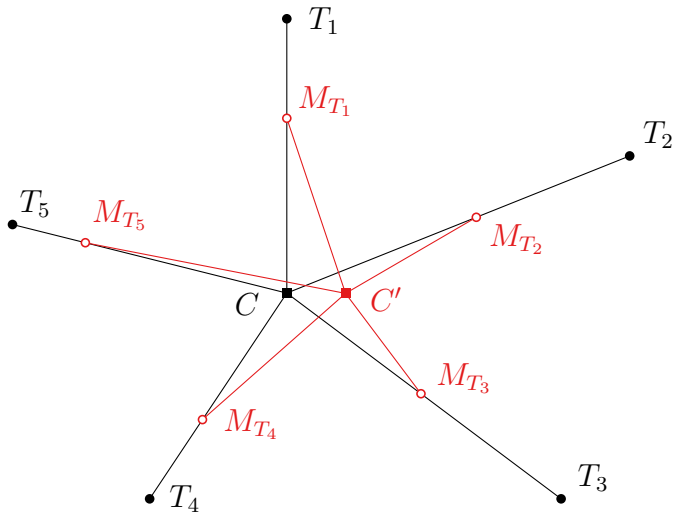


Figure 3 A visualization of the algorithm in Section 3.2.

4 Representative-set-based approach

Additionally, to reduce the computational cost of the finding central triangulation C, we utilize a *weighted representative-set-based approach*. This method is motivated by the concept of 1-*median coresets* [4], leveraging the fact that the parallel flip distance satisfies the triangle inequality. By discretizing the metric space, we construct a compact surrogate set $\mathcal{T}_{\mathrm{rep}}$ that preserves the properties of the objective function while significantly reducing the number of triangulations to be processed.

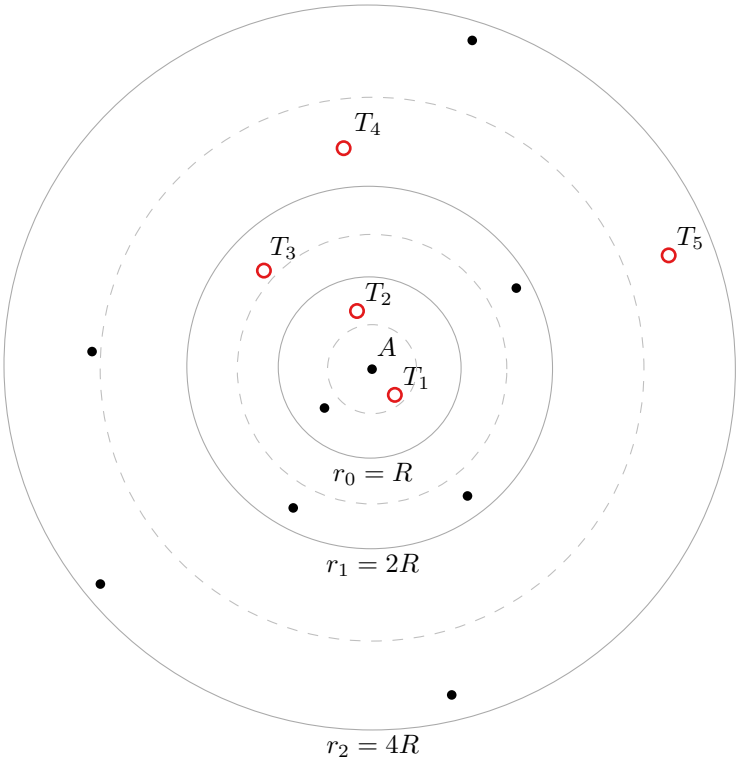


Figure 4 A visualization of the representative-set-based approach. There are $|\mathcal{T}| = 14$ triangulations. Here $A \in \mathcal{T}$, $\varepsilon = 0.5$, $\mathcal{T}_{\text{rep}}$ is $\{T_1, T_2, T_3, T_4, T_5\}$ with $w(T_1) = 2$, $w(T_2) = 2$, $w(T_3) = 4$, $w(T_4) = 1$, and $w(T_5) = 5$.

The construction of $\mathcal{T}_{\text{rep}}$ is executed through a *two-stage distance-based partitioning* process. First, if a central triangulation C from an initial solution is available, we set the *anchor triangulation* A as C; otherwise, we choose A as a random triangulation $T \in \mathcal{T}$. For each triangulation $T_i \in \mathcal{T}$, we compute $\pi(T_i, A)$ and the distance $\delta_i := |\pi(T_i, A)|$. We establish exponentially growing intervals that double in size based on the baseline scale $R := \sum_i \delta_i / |\mathcal{T}|$, and then partition $\mathcal{T}$ by grouping each triangulation T_i according to which interval its distance δ_i falls into. Next, we refine these partitions by sorting the triangulations by δ_i within each interval and further subdividing them into $1/\varepsilon$ subsets according to the granularity parameter $\varepsilon \in (0, 1)$. Finally, from each refined subset, we select a random triangulation and add it to $\mathcal{T}_{\text{rep}}$. We assign its weight $w(T)$ equal to the size of that subset, ensuring the total sum of weights equals the original size $|\mathcal{T}|$. See Figure 4.

The effectiveness and accuracy of this approach are summarized in Table 1. As shown, the proposed method substantially reduces the number of triangulations $|\mathcal{T}|$ – especially for large instances – while yielding objective values closely approximating the solutions derived from the whole triangulation set.

5 Results and discussions

In the following, we present a few observations about our approaches.

- When comparing the total parallel flip distance of *rirs-* instance and the three instances between top teams, we were in second place. See Table 2.

Table 1 Performance of the representative-set-based approach ($\varepsilon = 0.1$). Reduction denotes $|\mathcal{T}|$ decrease; objective ratio compares the results using $\mathcal{T}_{\text{rep}}$ to full $\mathcal{T}$.

Category	Number of instances	Avg. $\lvert\mathcal{T}\rvert$ reduction	Obj. value ratio
Small ($\lvert\mathcal{T}\rvert \leq 10$)	176	15.12%	1.082
Large ($\lvert\mathcal{T}\rvert > 10$)	74	78.35%	1.103
Total average	**-**	**33.84%**	**1.088**

- To understand above results, we compared the initial algorithm and various optimizations. For *rirs-* instances, the initial algorithm produced sufficiently short paths. For other instances, however, the path optimizations in Section 3.1 significantly reduced the distances in most cases. See Figure 5.
- For some *rirs-* instances, the large scale neighborhood search in Section 3.2 repeatedly generated the same central triangulation (i.e. $C = C'$ in Figure 3), regardless of the randomly selected M_T's. Nevertheless, the parallel flip paths were consistently shorter.

Table 2 The difference between four best teams' parallel flip distances and best among them. For example, our total parallel flip distance in *rirs-* instances is 3491 greater than the best 32897, meaning that 49 instances are on average 71.24 longer than the best. The number of instances of each type (*random-, woc-, rirs-*) are (100, 101, 49).

instances	best sum	Shadoks	ETH Flippers	ours	jflip2
random-	3221	1(0.01)	66(0.66)	132(1.32)	85(0.85)
woc-	3434	0(0.0)	37(0.37)	90(0.89)	82(0.81)
rirs-	32897	0(0.0)	4774(97.43)	3491(71.24)	3840(78.37)
Total	39553	1	4877	3713	4007

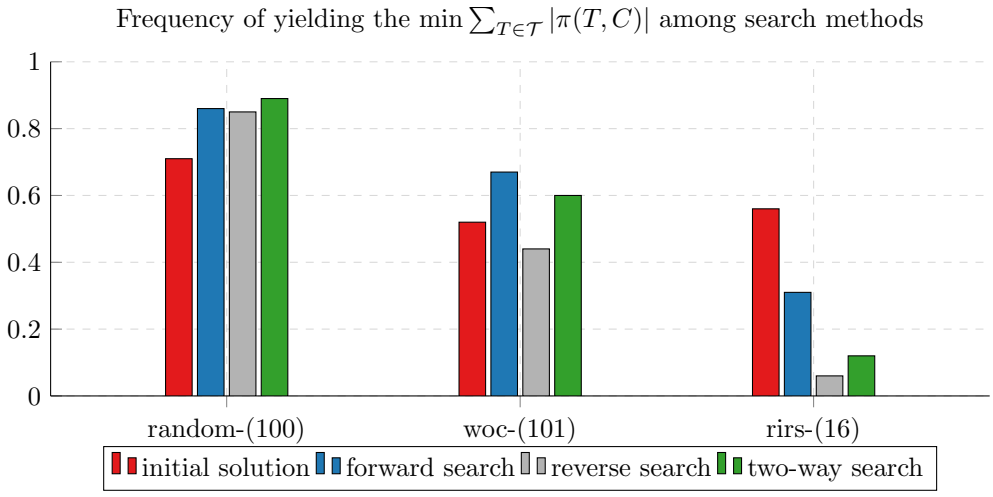


Figure 5 To evaluate the initial algorithm against various path optimizations, we measured the frequency of achieving the min $\sum_{T\in\mathcal{T}} |\pi(T,C)|$ among them from our initial solutions. We tested *rirs-* instances with $|\mathcal{T}| = 20$ and all *random-* and *woc-* instances.

References

1 Oswin Aichholzer, Joseph Dorfer, Sándor P. Fekete, Phillip Keldenich, Peter Kramer, and Stefan Schirra. Central Triangulation under Parallel Flip Operations: The CG:SHOP Challenge 2026, 2026. arXiv:2603.18812.

2 Jérôme Galtier, Ferran Hurtado, Marc Noy, Stéphane Pérennes, and Jorge Urrutia. Simultaneous edge flipping in triangulations. *International Journal of Computational Geometry & Applications*, 13(02):113–133, 2003. doi:10.1142/S0218195903001098.

3 Sabine Hanke, Thomas Ottmann, and Sven Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2(8):570–579, 1996. doi:10.3217/JUCS-002-08-0570.

4 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th annual ACM Symposium on Theory of Computing (STOC)*, pages 291–300, 2004. doi:10.1145/1007352.1007400.

5 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22:333–346, 1999. doi:10.1007/PL00009464.

6 Iyad Kanj, Eric Sedgwick, and Ge Xia. Computing the flip distance between triangulations. *Discrete & Computational Geometry*, 58:313–344, 2017. doi:10.1007/S00454-017-9867-X.

7 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. doi:10.1016/J.COMGEO.2014.11.001.